

Glue Framework

개발자 Guide

Document Release 1.1

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

목 차

Glue Framework	1
Document Release 1.1	1
1. 서론	9
1.1 본 문서의 목적	9
1.2 범위	9
1.3 용어	9
2. Glue Framework 아키텍처	10
2.1 시스템 구성도	10
2.1.1 Presentation Layer	10
2.1.2 Business Layer	10
2.1.3 Persistent Layer	11
2.1.4 Integration Layer	11
2.2 Glue Framework 동작 원리	11
2.2.1 Web Framework Wrapper	12
2.2.2 Service 처리 부분	13
2.2.3 Non-UI 처리 부분	14
3. 개발환경 설정	16
3.1 IDE 설치	16
3.1.1 Glue Plug-In 설치	17
3.1.2 GlueSDK 설치	19
3.1.3 기타	20
3.2 WAS 환경설정	22
3.2.1 Tomcat 7	24
3.2.2 Weblogic 12C	25
3.2.3 JBoss EAP 6	26
3.3 Glue Project	27
3.3.1 Glue Java Project	29
3.3.2 Glue Maven Project	29
3.4 Glue Activity Diagram	30
3.4.1 Activity	33
3.4.2 Reuse Activity	35
3.4.3 서비스명세서	36
3.5 Glue Query Editor	36
4. GlueContext	39
5. Service	41
5.1 transaction-manager	42
5.2 Activity 구현	43
5.2.1 Activity Parameter 이용	44
5.2.2 DAO Class 의 Handling	45
5.3 Reuse Activity	46
5.3.1 Jdbc Search Activity	46
5.3.2 Jdbc Modify Activity	47
5.3.3 GlueParseMessage	50
5.3.4 GlueCreateMessage	51

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

5.3.5	Routing Activity	52
5.3.6	Sub Service 이용 (Service 재사용)	52
6.	Data Access Object(DAO)	54
6.1	GlueGenericDao	54
6.2	GlueJdbcDao	55
6.2.1	Glue Query File	57
6.2.2	GlueParameter	59
6.2.3	GlueColumnManager	59
6.2.4	Reuse & Custom Activity	60
6.3	GlueMyBatisDao	60
6.3.1	Mapper XML 파일	61
6.3.2	Parameters/Result Maps	63
6.3.3	GlueParameter	64
6.3.4	Reuse & Custom Activity	65
6.4	GlueHibernateDao	68
6.4.1	hbm.xml	70
6.4.2	Glue Query File(HQL)	72
6.4.3	GlueHiberanteParameter	74
6.4.4	Reuse & Custom Activity	75
7.	Cache 활용	80
7.1	정의 및 역할	80
7.2	사용 목적	80
7.3	Caching Solution 소개 및 비교	80
7.4	Glue Caching Component Architecture Diagram	80
7.5	JCS Cache 개발 가이드	81
7.5.1	Remote Cache	83
7.6	EhCache 개발 가이드	86
7.6.1	EhCache	86
7.6.2	Spring 에서의 EhCache 활용	87
7.6.3	Glue Framework 에서의 EhCache 활용	87
7.6.4	Business 적용	87
7.6.5	EhCache 의 구조	87
7.6.6	EhCache 설정	89
8.	Logging	90
8.1	정의 및 역할	90
8.2	작동 방식	90
8.3	개발 가이드	90
8.3.1	Log4J 를 사용하는 경우	91
8.3.2	JDK1.4 Logger 를 사용하는 경우	91
8.3.3	Logging Component 사용 예제	92
9.	Exception 처리	94
9.1	정의 및 역할	94
9.2	Checked Exception vs Unchecked Exception	94
9.3	Glue Framework 에서의 Exception 처리	95
9.3.1	Exception 처리 전략	95
9.3.2	각 Layer 별 Exception 처리	95
9.3.3	GlueException 계층도	96
9.3.4	GlueDaoException 처리 방법	97

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

10. Web 화면 개발	98
10.1 DispatcherServlet 정의	98
10.2 HandlerMapping 설정	98
10.2.1 BeanNameUrlHandlerMapping	98
10.2.2 SimpleUrlHandlerMapping	98
10.3 ViewResolver 설정	99
10.3.1 ViewResolver 역할	99
10.3.2 ViewResolver 종류	99
10.4 Controller	100
10.4.1 GlueAbstractController 사용	100
10.4.2 GlueSimpleController	100
10.4.3 Exception 발생 시 View Page(GlueAbstractController)	100
10.5 Interceptor	100
10.5.1 Interceptor 구현	100
10.5.2 Interceptor 설정	101
11. Non-UI 개발	103
11.1 Glue Framework 설정	103
11.1.1 Glue Non-UI 환경파일 구성	103
11.1.2 Glue Framework 설정	103
11.2 설계서 및 설계 산출물	103
11.2.1 Activity Diagram	103
11.2.2 Service XML 파일	104
11.2.3 Query 파일	104
11.2.4 Message Layout XML 파일	105
11.2.5 Message Layout XML 파일과 Message Data Handling	106
11.3 개발 범위 및 방법	108
11.3.1 개발 범위	108
11.3.2 개발 방법	109
12. Globalization & Multi TimeZone	110
12.1 Globalization	110
12.1.1 Glue Framework 설정	110
12.1.2 사용 방법	110
12.2 Multi TimeZone	111
12.2.1 사용 방법	111
12.3 User Locale & TimeZone 설정	112
12.3.1 적용 예	112
13. Excel	115
13.1 Excel Export	115
13.1.1 GlueExcelInfo	115
13.1.2 GlueXMLViewController	115
13.1.3 excel-views.xml	116
13.2 Excel Import	116
13.2.1 GlueXMLViewController	116
13.2.2 Glue Service	116
13.3 Glue 환경 설정	117
13.4 Sample 화면	118
13.5 Customizing	119
13.5.1 POI 용 View 제작	120
13.5.2 JExcel View 제작	120

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

14. 웹서비스	121
14.1 JAX-WS	121
14.1.1 Web Service 작성	122
14.1.2 Server 부분	123
14.1.3 Client 부분	124
14.1.4 Glue 에서 제공하는 WebService 클래스	125
14.2 RESTful	126
14.2.1 Server 파트	127
14.2.2 Client 파트	129
14.2.3 Glue 에서 제공하는 RESTful 용 Controller	130
15. Scheduler	132
15.1 Spring Scheduler	132
15.1.1 GlueTaskScheduler	133
15.1.2 Task 실행 History	134
15.2 Quartz Job Scheduler	135
15.2.1 GlueQuartzJobBean	139
15.2.2 Job 실행 History	139
15.3 Scheduler Luanch	141
15.3.1 Standalone JVM	141
15.3.2 서블릿 컨테이너	141
15.3.3 Glue Scheduler Server	142
16. Spring Mobile	145
16.1 spring mobile 라이브러리 설정	145
16.2 interceptor	145
16.2.1 DeviceResolverHandlerInterceptor	146
16.2.2 SiteSwitcherHandlerInterceptor	147
16.3 WURFL	147
17. AUDIT	150
17.1 Audit 기능 개요	150
17.2 Audit 설정	150
17.2.1 Audit 항목 설정(glue.properties)	150
17.2.2 GlueContext 에 GlueAuditAttributes 설정	150
17.2.3 GlueParameter 에 auditAttributes 설정	151
17.2.4 SQL 작성	151
18. 기타	153
18.1 glue-config.xml	153
18.2 applicationContext.xml	154
18.3 web.xml	155
18.3.1 Spring MVC	157
18.3.2 Struts 1	157

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

그림 목차

[그림 1] 아키텍처.....	10
[그림 2] Web 동작원리	12
[그림 3] Activity Diagram.....	14
[그림 4] Activity Diagram.....	15
[그림 5] Eclipse Download	16
[그림 6] Eclipse 설치 결과	17
[그림 7] Plug-In 설치	17
[그림 8] 바로가기 -clean 적용	18
[그림 9] GlueSDK 설치	19
[그림 10] SDK Location 설정	19
[그림 11] web module structure	23
[그림 12] catalina.bat	24
[그림 13] Tomcat	25
[그림 14] setDomainEnv.cmd	25
[그림 15] WebLogic	26
[그림 16] standalone.conf.bat.....	26
[그림 17] JBoss	27
[그림 18] JBoss Console	27
[그림 19] Glue Project 생성 wizard.....	28
[그림 20] Glue Project 구조.....	28
[그림 21] pom.xml 의 Dependency Hierachy	29
[그림 22] Activity Diagram.....	30
[그림 23] Glue Activity Diagram 생성 wizard	31
[그림 24] Glue Activity Diagram.....	31
[그림 25] Pallete 의 Layout.....	32
[그림 26] Activity Diagram 의 Outline 뷰	33
[그림 27] Activity 정렬.....	33
[그림 28] Activity Property Edior	34
[그림 29] View 형태의 Activity Property Editor	34
[그림 30] Dialog 형태의 Activity Property Editor	35
[그림 31] Activity Setting 의 버튼	35
[그림 32] Reuse Activity List.....	36
[그림 33] Activity Diagram 의 Description	36
[그림 34] Glue Query Editor	37
[그림 35] Glue Query Editor 생성 wizard	37
[그림 36] Query JBDC Connection.....	37
[그림 37] Test Query	38
[그림 38] GlueContext 의 Class 관계	39
[그림 39] 내부저장소.....	40
[그림 40] DefaultRouter	52
[그림 41] Data Access Object	54
[그림 42] GlueJdbcDao 의 bean 들과의 관계	56
[그림 43] GlueMyBatisDao 의 bean 들과의 관계	60
[그림 44] GlueHibernateDao 의 bean 들과의 관계	69
[그림 45] CacheManager.....	81
[그림 46] Remote Cache Server	85
[그림 47] Remote Cache Server	85
[그림 48] 1 번 서버에서 Put Cache	86
[그림 49] 2 번 서버에서 Get Cache	86
[그림 50] Ehcache	88

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

[그림 51] EhCache	88
[그림 52] EhCache	88
[그림 53] Logger	90
[그림 54] Checked Exception vs Unchecked Exception	94
[그림 55] 각 Layer 별 Exception 처리	95
[그림 56] Exception	96
[그림 57] Activity Diagram	104
[그림 58] Query	105
[그림 59] MessageLayout	106
[그림 60] MessageParse	107
[그림 61] MessageParse	108
[그림 62] Interceptor 구현	113
[그림 63] Excel Export	115
[그림 64] Excel Import	116
[그림 65] excel-service.xml	118
[그림 66] Default Router 분기	130
[그림 67] Cron Expression	133
[그림 68] ERD	135
[그림 69] Glue Scheduler Conceptual Diagram	136
[그림 70] ERD	140
[그림 71] java Usage	142

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

표 목 차

[표 1] GlueF/W 관련 라이브러리 정리	22
[표 2] GlueJdbcSearch.....	47
[표 3] GlueJdbcModify.....	48
[표 4] GlueMessageParse	50
[표 5] GlueCreateMessage.....	51
[표 6] GlueSubService.....	53
[표 7] new-transaction 설정에 따른 commit, rollback 범위	53
[표 8] GlueGenericDao 메소드.....	55
[표 9] GlueGenericJdbcDao 메소드	56
[표 10] GlueJdbcDao 의 property.....	57
[표 11] GlueJdbcDao 에서 사용 가능한 SQL 유형	57
[표 12] GlueGenericHibernateDao 메소드	68
[표 13] Caching solution.....	80
[표 14] GlueDaoException	97
[표 15] ViewResolver	100
[표 16] Layout	103
[표 17] <jaxws:endpoint/> tag 의 필수 속성값	123
[표 18] GlueJobScheduler 의 property	137
[표 19] Trigger 의 Property	138
[표 20] JobDetail 의 Property.....	139
[표 21] SiteSwitcherHandlerInterceptor 의 factory-method.....	147

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

1. 서론

1.1 본 문서의 목적

이 문서는 Glue Framework 을 이용하여 개발시 Glue F/W 이 제공하는 기능을 설명하며 더불어 Glue Framework 을 이용하여 연결하고자 외부 요소(예를 들어 Mybatis, Hibernate, log 등) 들의 연계방법에 대해 설명한다.

이 문서는 Glue F/W 설치, 재사용 가능한 Activity 별 사용법, Cache, Logging 등 Glue F/W 제공하는 기능들에 대해 설명하고 필요한 경우 예제를 포함한다.

각 기능별 완성된 예제는 튜토리얼 셋(set)으로 완성된 예제 프로젝트와 함께 별도 튜토리얼 가이드로 제공한다

1.2 범위

이 문서는 Glue Framework 에서 사용되는 Struts Framework, Spring MVC Framework 등 외부 요소에 대한 연관 관계를 설명하는데 초점이 맞춰져 작성되었으며 Struts, Spring MVC Framework 등 외부 요소에 대해 직접 기술하지는 않는다.

Glue Framework 이 예제 작성을 위해 이용한 JQuery 등 화면 요소에 대해서는 이 문서에서는 다루지 않는다.

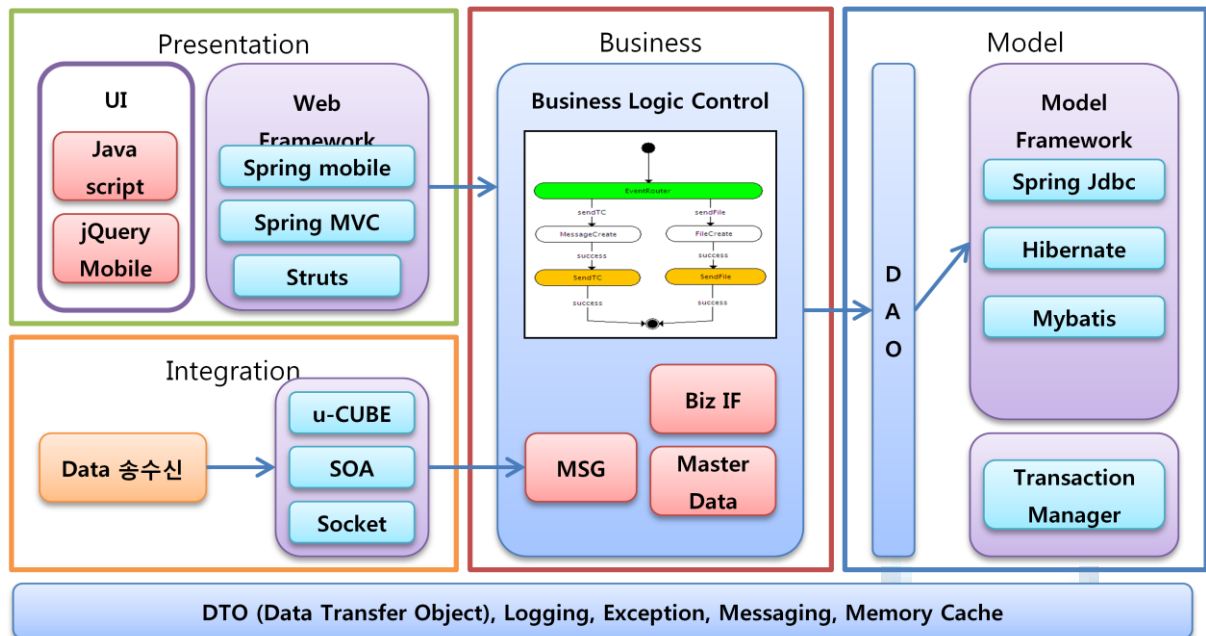
Glue F/W 는 특정 Web Application Server(WAS), Database 에 종속적이지 않으므로 본 문서의 예제는 개발자들이 많이 이용하는 WAS, DB 를 기준으로 작성하였으며 특정 제품에 대한 선호도는 없음을 이 문서에서 밝힌다.

1.3 용어

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

2. Glue Framework 아키텍처

2.1 시스템 구성도



[그림 1] 아키텍처

2.1.1 Presentation Layer

Presentation Layer 는 화면(UI)와 Controller(Web F/W) 로 나눈다.

1. UI(화면) : UI 는 JSP, MiPlatform, Gauge, Trustform, Flex, JQuery, JQuery Mobile 등을 활용하여 개발할 수 있으므로 개발자(혹은 프로젝트)가 이용하기 위한 편한 화면 프로그래밍 방식을 선택한다. Glue F/W 에서 선호 / 추천하는 프로그래밍 방식은 없다. 다만 편의를 위해 특정 회사의 요청으로 특정 화면 프로그래밍을 위한 어댑터(MiPlatform, Trustform 등)을 제공한다.
2. Controller : Glue F/W 4.0 Controller 부분은 Spring MVC 를 채택하고 있다. (Glue F/W 3.x 대 사용되었던 Struts 기능도 함께 제공한다. 다만 본 문서를 비롯 Glue F/w 4.0 과 관련된 모든 산출물이 Spring MVC 를 이용하여 개발되었다.) Controller 부분은 Business Service 를 연결하는 부분을 구현 제공하며, Controller Classs 가 이 부분에 해당된다. Controller 가 하는 일은 화면의 Flow 를 Control 하고 Glue Framework 의 Service 를 호출 하는 역할을 한다. 또한 Glue Framework 은 화면을 Control 하는 다른 MVC Framework(Struts 등)에 대한 구현을 제공하여 Glue Framework 을 이용하는 개발자 및 설계자가 이 부분에 대해 추가로 개발 및 Architecture 를 수립할 비용을 감소 시켜 준다.

2.1.2 Business Layer

Glue Framework 에서 Service 부분에 해당하는 부분으로 UI 부분과 더불어 개발자가 비즈니스 로직에 맞추어 개발해야 하는 부분이다.

	문서명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Business Layer 는 POJO(Plain Old Java Object)로 구현 되어야 한다. Glue Framework 의 Service 는 모두 POJO 로 구현되어 있고 Model Layer 와 Presentation Layer 와의 모든 연결은 Interface 로 구현되고 Data 또한 Glue 의 통합 Context 를 통해 Framework 의 Coupling 을 최소화 하였다.

개발자는 Model Layer 혹은 Presentation Layer 의 변경이 발생하더라도 Business Layer 의 코드 변경은 최소화 되어야 하기 때문에 Business Layer 를 구현할 때 Model Layer 나 Presentation Layer 의 어떠한 Class 및 Interface 도 가져다 구현하지 않아야 한다.

2.1.3 Persistent Layer

Model Layer 의 Framework 을 Implements 하여 제공 하고 표준 Interface 만을 제공하여 Coupling 을 최소화 를 제공한다.

DB 를 Handling 하는 부분으로 모든 DB 를 동일 하게 처리 하기 위해 JDBC 표준으로 구현되어 있다.

모든 Data Handling 하는 부분을 Interface 화 하여 Model Layer 의 Framework 이 변경되어도 Business Layer 의 Service Code 는 변경이 최소화 되게 하는 역할을 담당한다. (DAO Class)

이를 위하여 개발자는 Glue Framework 에서 제공되는 DAO Class 를 통해 Data Handling 을 하여야 한다.

2.1.4 Integration Layer

외부의 다른 시스템과 데이터를 송수신하기 위한 부분으로 기본적인 HTTP 통신을 비롯 웹서비스(JAX-WS, RESTful)을 이용한 형태, u_CUBE 와 같은 EAI 와 송수신을 위한 인터페이스를 제공하여 개발자가 간단한 셋팅만으로 별도의 개발과정 없이 이용할 수 있도록 제공한다.

2.2 Glue Framework 동작 원리

화면의 경우 해당 URL 에 의해 WEB xml 의 <servlet-Mapping> tag 에 정해진 dispatcher 로 분기되어 진다. 현재 설명된 예제에서는 "MVC"라는 URL 확장자에 의해 해당하는 servlet mapping 은 "dispatcher" 으로 "org.springframework.web.servlet.DispatcherServlet" class 를 이용하게 된다.

	문서명	Glue Framework 개발자 가이드
	제품버전	V 4.0

<http://localhost:8080/sample-ui/sample.mvc>

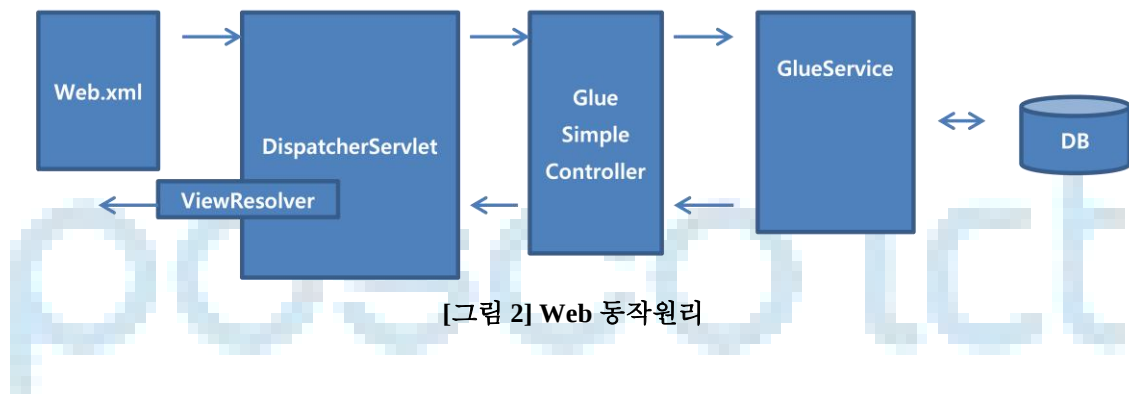


```

<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>
    org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>*.mvc</url-pattern>
</servlet-mapping>

```

Web.xml



[그림 2] Web 동작원리

2.2.1 Web Framework Wrapper

2.2.1.1 SpringMVC 의 Controller 를 확장

Aga

1. JSP 로 부터 Submit 이 호출 되면 Spring Framework 의 해당 Controller 가 실행 된다. 모든 Activity type(Class)는 Glue Framework 에서 제공되는 Action Class 를 상속 받은 Class 이어야 한다.
2. JSP 는 반드시 자신의 Service 를 지정하여야 한다. 지정 방법은 Hidden Tag 로 다음과 같이 지정하여야 한다.

<Sample -JSP>

```
<input type="HIDDEN" name="ServiceName" value="EmpService"/>
```

GlueSimpleController Class 는 Spring Framework 에서 Glue Framework 을 호출하기 위한 Wrapper Class 로서 다음과 같은 역할을 수행한다.

1. Glue Context 생성

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

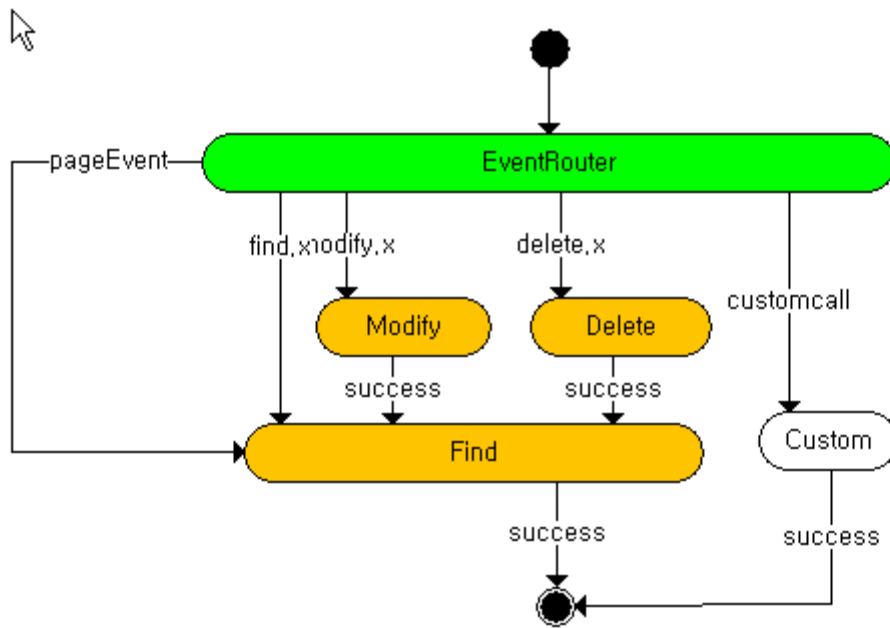
- GlueContext 는 Request 마다 생성되는 Data Transfer Object 이다. GlueContext 는 JSP 부터 Activity 까지 모든 Data 의 공유 통로이고 Web 에서 GlueContext 는 Request, Session, Attribute Data 까지 포함한다.
 - GlueContext 는 Spring Framework 의 Application Context 를 Loading 하고 Bean Object 를 서비스 한다. 개발자는 public static GlueBeanFactory getBeanFactory() Method 를 통해 Application Context 의 Bean 을 획득하여 이용할 수 있다. 대표적인 사례는 DAO Class 를 이용하는 것이다.
 - glue.properties 의 Loading 및 Property 에 대한 Access 제공
 - request,session Data 를 GlueContext 에 저장
2. GlueBizController 호출
- GlueBizController 는 GlueContext 를 Parameter 로 호출되어 진다. GlueBizController 는 Request 에서 “<input type="HIDDEN" name="ServiceName" value="EmpService"/>” 를 찾아와 해당 Service 를 Loading 하게 된다. 즉 GlueContext 에서 ctx.get(“ServiceName”);으로 Return 된 “EmpService”를 Memory 에서 찾고 없으면 EmpService.xml 파일을 읽어서 Loading 하게 된다.
3. View Page Control 기능
- Activity 에서 “ctx.put(GlueWebConstants.VIEW_PAGE,"showDept");” 과 같이 View Page 를 지정하였다면 GlueController 에서는 해당 Page 를 View Page 로 분기하여 준다.
4. Exception 처리
- Service 실행 중 Exception 이 발생하였을 경우 servlet.xml 에서 해당 Controller 에 대한 errorPage 가 정의 되었다면 errorPage 가 View Page 로 설정되며 errorPage 가 정의 되어 있지 않다면 일반 설정(viewPage)을 따르게 된다.
 - Interceptor 설정을 통해서 별도의 예러 처리 로직을 구현 할 수도 있다

2.2.2 Service 처리 부분

Service 는 하나의 Transaction 으로 처리되는 Business 단위로 구성된다.

모든 개발 부분은 Service 의 Activity 로 구현되고 그 결과는 GlueContext 로 공유된다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 3] Activity Diagram

1. GlueBizController 에서 GlueContext 에 있는 ServiceName 을 가지고 GlueServiceLoader 를 통해 Service 를 조회 한다
2. GlueServiceLoader 는 Cache 에서 해당 Service 를 조회 하고 없으면 ServiceName + .xml 파일을 Loading 한다
3. GlueBizController 는 Service.xml 내용을 분석하여 각각의 Activity Class 들을 호출한다.
4. GlueBizController 는 Service 에서 “<transaction-manager id="tx1" commit="true"/>” 로 등록되어 있는 Transaction 을 Commit 또는 Rollback 하게 된다.
5. Logging 처리 : Controller 는 Service 의 수행 시간과 Activity 의 수행시간 Exception 의 내용등을 Logging 한다

2.2.3 Non-UI 처리 부분

NonUI 는 시작이 Message 또는 File Message 로부터 수신되어 Framework 의 Service 를 처리한다.

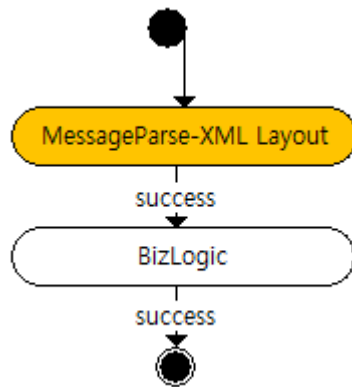
Message 수신을 위한 별도의 Servlet(HttpReceiver)을 이용하며, 다음과 같이 web.xml 에 정의한다.

```

<servlet>
  <servlet-name>NonUIReceiver</servlet-name>
  <servlet-class>
    com.poscoict.glueframework.web.GlueHttpReceiverAdapter
  </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>uCubeReceiver</servlet-name>
  <url-pattern>*.msg</url-pattern>
</servlet-mapping>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 4] Activity Diagram

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

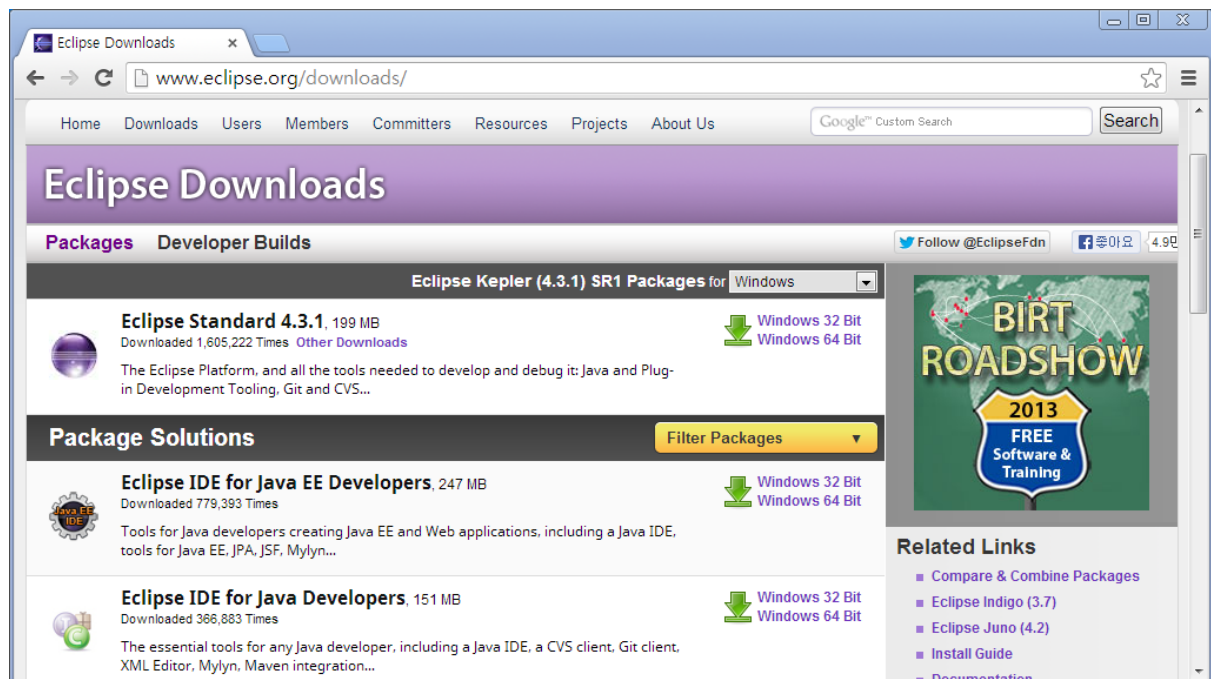
3. 개발환경 설정

이번 장의 내용을 응용해서 프로젝트에 맞는 개발 환경 구성할 수 있습니다.

기본적으로 Java SE 6 이상의 환경이어야 하며, Java 설치 내용은 다루지 않습니다.

3.1 IDE 설치

통합개발환경(IDE, Integrated development environment)으로 Eclipse 를 사용하며, Eclipse 사이트(<http://www.eclipse.org>)에서 Juno 버전 이상의 Eclipse IDE for Java EE Developers 또는 Eclipse IDE for Java Developers 패키지를 다운받아 사용하도록 한다.



[그림 5] Eclipse Download

Eclipse 는 압축을 해제하는 것으로 설치가 된다. 이후, Eclipse 설치 폴더는 ECLIPSE_HOME 으로 지칭한다.

Eclipse 설치 경로인 ECLIPSE_HOME 에는 한글이나 공백이 포함되지 않는 것이 좋다. 이후 workspace 를 지정할 때도 한글이나 공백이 포함되지 않도록 한다.

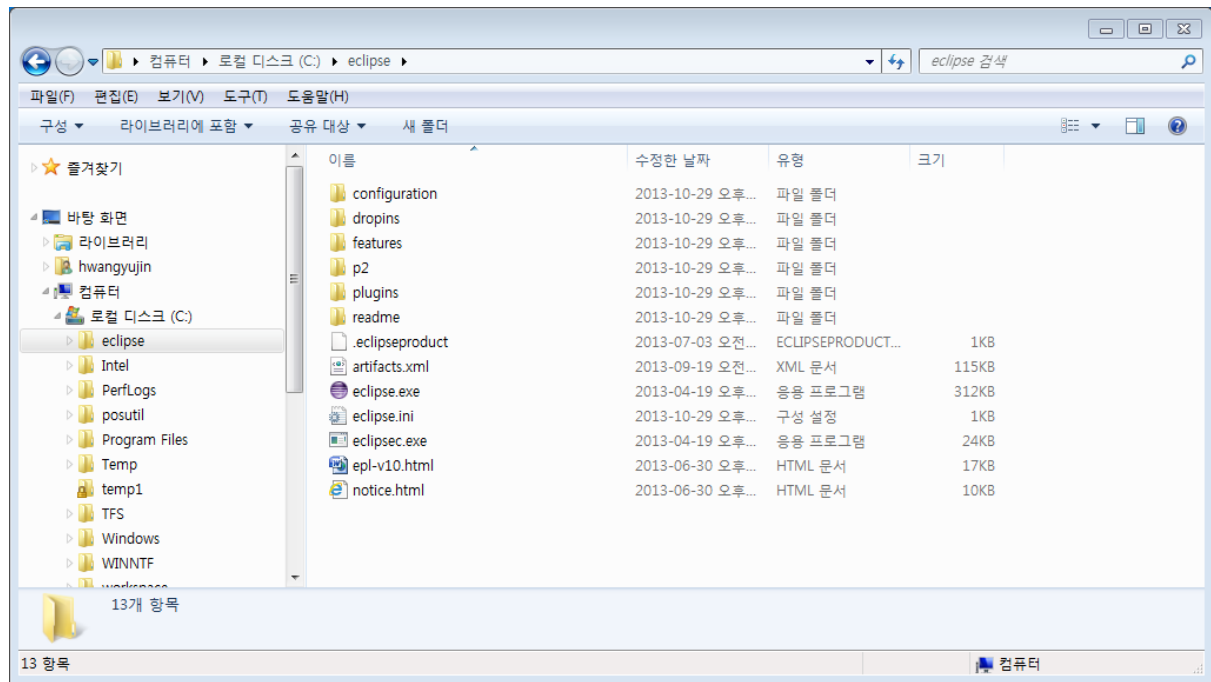
Eclipse 설치 후, 실행파일을 통해 Eclipse 가 잘 설치 되었는지 확인한다. 최소 실행시 workspace 를 묻는 창이 나타나며, 그 위치는 ECLIPSE_HOME/workspace 로 지정하도록 한다.

- 실행파일 : {ECLIPSE_HOME} / eclipse.exe
- Glue Plug-In 설치 위치 : {ECLIPSE_HOME} / dropins
- workspace 폴더위치 : {ECLIPSE_HOME} / workspace
- users 폴더 위치 : {ECLIPSE_HOME} / users

users 폴더는 이후 GlueSDK 를 설치할 위치이다.

다음은 Eclipse 를 압축 해제한 모습입니다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



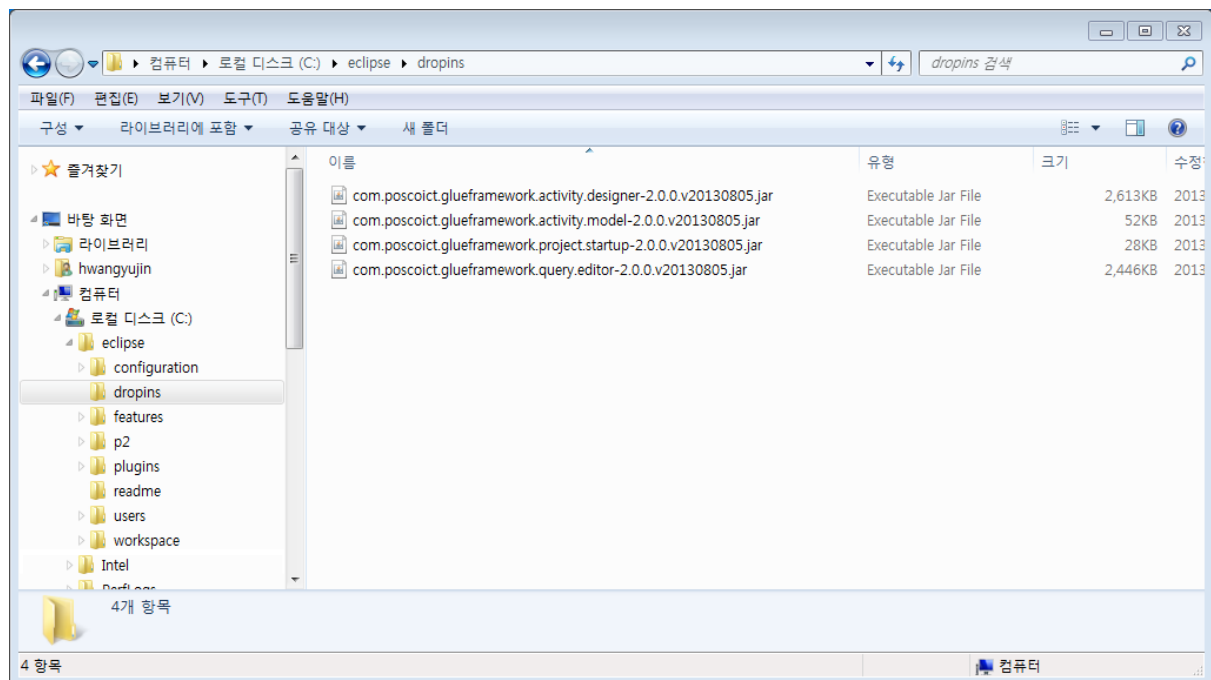
[그림 6] Eclipse 설치 결과

Eclipse IDE 를 설치한 후에는 Glue Framework 과 관련해서 다음 설치 과정을 진행한다.

1. Glue Plug-In 설치
2. GlueSDK 설치

3.1.1 Glue Plug-In 설치

Glue Plug-In 을 설치 폴더에 복사한다.



[그림 7] Plug-In 설치

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

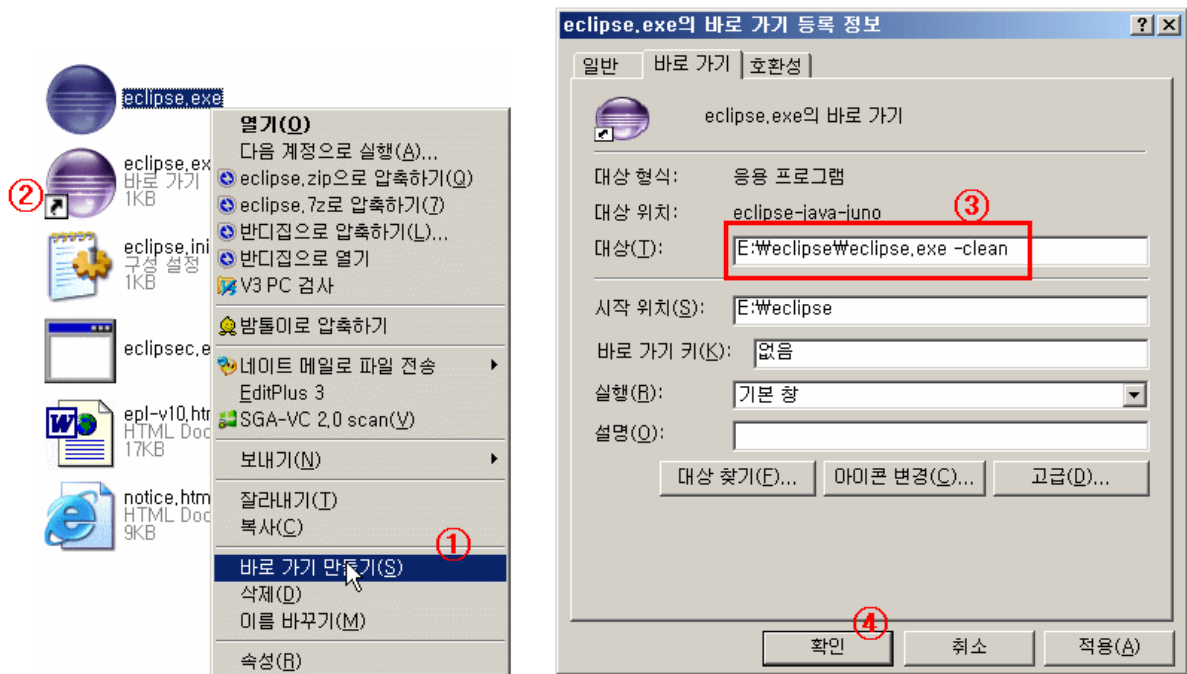
Glue Plug-In 의 설치는 다음 4 개 파일을 ECLIPSE_HOME/dropins 에 복사하면 된다.

- com.poscoict.glueframework.activity.designer-{version}.jar
- com.poscoict.glueframework.activity.model-{version}.jar
- com.poscoict.glueframework.project.startup-{version}.jar
- com.poscoict.glueframework.query.editor-{version}.jar

Glue Plug-In 의 패치/업그레이드는 2 가지 방법을 선택해서 수행할 수 있다.

- "-clean" 옵션 이용
- 재설치(삭제후 설치)

첫번째 방법인 Clean 옵션을 이용하는 방법은 바로가기를 생성해서, 그림과 같이 바로가기 등록정보의 대상 프로그램에 -clean 옵션을 주는 것이다. ECLIPSE_HOME / dropins 에서 4 개 파일을 변경하고, "-clean" 옵션이 포함된 바로가기를 통해 Eclipse 를 실행하면 된다.



[그림 8] 바로가기 -clean 적용

두번째 방법은 Glue Plug-In 재설치 절차를 따르는 것으로, 그 절차는 다음과 같다.

1. 실행중인 Eclipse 종료.
종료전 Activity Diagram 및 Query Editor 파일은 닫아 준다.
2. 기존 plugin 삭제 후 Eclipse 실행.
Glue Plug-In 이 삭제되었음을 Eclipse 가 인지하게 한다.
3. Eclipse 종료
Eclipse 가 종료된 상태에서 새로운 Plug-In 을 복사한다.
4. 새로운 plugin 복사 후 Eclipse 실행
5. GlueSDK SDK Location 재 설정 (Window → Preferences → Glue Framework)

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

3.1.2 GlueSDK 설치

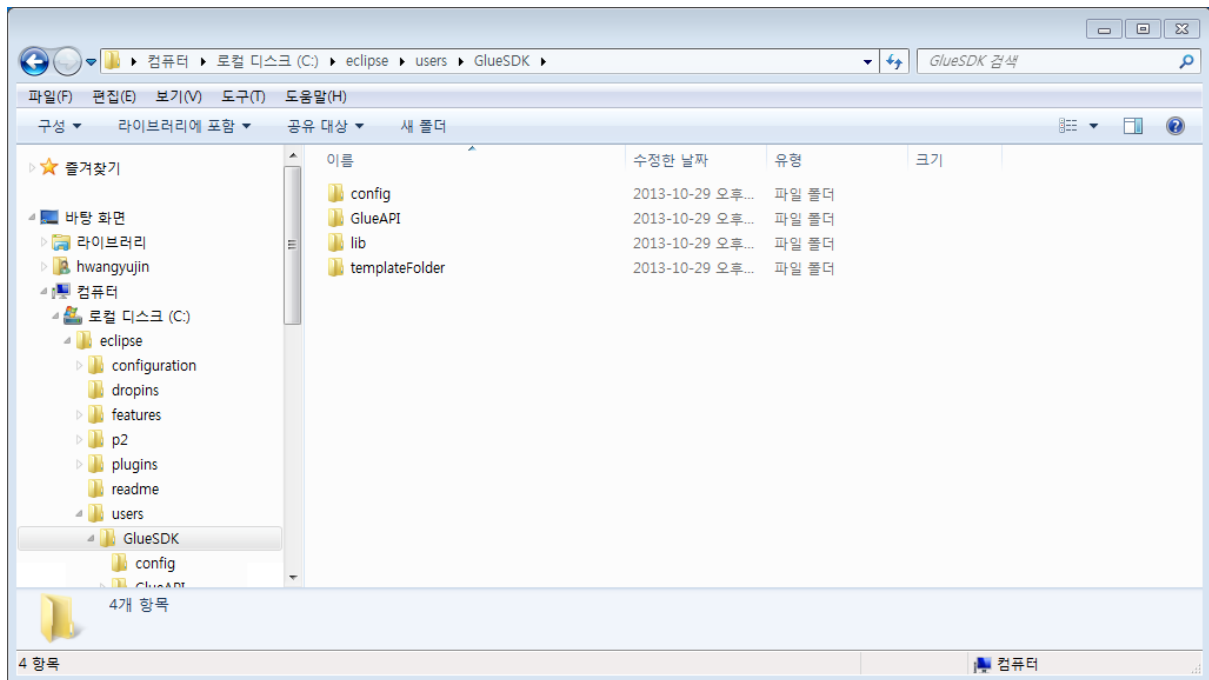
GlueSDK 란 개발에 필요한 모든 작업환경을 설정하고 Upgrade 자동화를 지원하는 툴이다. GlueSDK 의 내용은 개발자가 개별로 설정하지 않고, PL 의 의해 해당 프로젝트에 맞게 GlueSDK 를 구성하고, 전체 개발자가 동일 GlueSDK 를 사용하기를 권장한다.

GlueSDK 설치 과정은 다음과 같다.

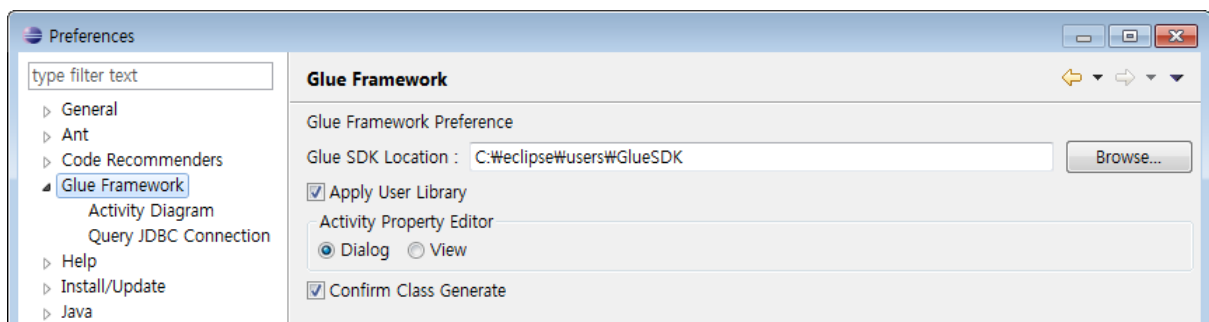
1. GlueSDK Download
2. users 폴더에 GlueSDK 압축 해제.
3. Glue SDK Location 지정
4. Apply 버튼을 클릭한다.

GlueSDK 는 PCoP (또는 메일)을 통해 압축 파일 형태로 배포된다.

IDE 설치시 users 를 폴더를 생성하지 않았다면 생성한다. 『GlueSDK 설치』 그림을 참고해서 다운받은 GlueSDK 설치(압축해제)한다. Eclipse 의 Window 메뉴에서 Preference 를 찾아 실행한다. 『Glue SDK Location 설정』 그림과 같은 Preference 창이 실행되며, 왼쪽에서 Glue Framework 을 찾아 SDK Location 에는 GlueSDK 설치 폴더를 지정한다.



[그림 9] GlueSDK 설치



[그림 10] SDK Location 설정

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Glue Framework Library 가 바뀌었다면, 즉 GlueSDK 가 Patch/Upgrade 되었다면, Eclipse 를 재실행하거나, Preferences 창에서 GlueSDK Location 을 재설정 하여야 한다. 『Glue SDK Location 설정』 그림과 같은 Preference 창에서 Apply 버튼을 눌러주면 된다.

3.1.3 기타

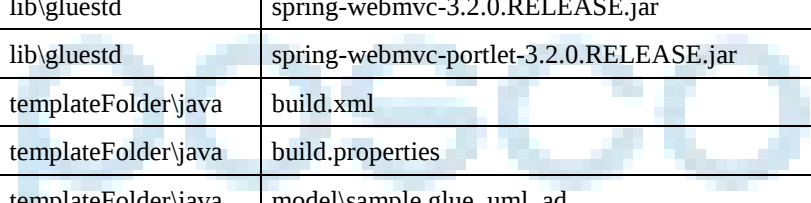
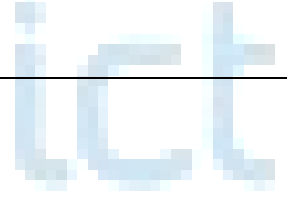
GlueSDK 에는 다음과 같은 4 가지 폴더로 나뉘어져 있다.

- config : config 폴더에는 glue-config.xml 파일이 있으며, Glue Plug-In 에서 사용하는 설정정보를 관리하고 있다.
- GlueAPI : GlueAPI 폴더에는 site-javadoc.zip 파일이 있으며, 프로젝트 정보가 문서화 되어 있다. 여기서 프로젝트는 Glue Framework 을 의미하며, Glue Framework 의 의존물 정보, 라이선스 등과 java doc 를 포함한다.
- lib : lib 폴더는 gluelib 와 gluestd 로 구분되며, 그외 폴더는 gluestd_{etc} 형태이다. Glue Plug-In 에서는 gluelib 와 gluestd 의 속한 library 만 User Library 로 자동 인식한다. gluelib 에는 Glue Framework 의 모듈들이 위치한다. glue-schema 모듈과 glue-core 모듈은 필수이며, 그외는 선택해서 사용한다. gluestd 에는 Glue Framework 의 각 모듈의 의존 library 들이 위치한다. gluestd 의 의존 library 들은 gluestd_{etc} 에서, 일부 의존 library 는 site 에서 직접 다운받아 사용자가 직접 재 구성해야 한다.
- templateFolder : templateFolder 에는 2 종류의 sample template 을 포함한다. Glue Plug-In 에서 Glue Project 생성시 java type 과 maven type 을 선택할 수 있으며, 이때 templateFolder 에 내용물이 사용된다.

다음은 GlueSDK 의 기본 구성물이다.

구분	파일명	설명
config	glue-config.xml	Eclipse Plugin 에서 참고하는 정보
GlueAPI	site-javadoc.zip	Java Doc
lib\gluelib	glue-core-{version}.jar	Glue Framework 의 module
lib\gluelib	glue-hibernate-{version}.jar	
lib\gluelib	glue-mybatis-{version}.jar	
lib\gluelib	glue-schedule-{version}.jar	
lib\gluelib	glue-schema-{version}.jar	
lib\gluelib	glue-tester-{version}.jar	
lib\gluelib	glue-ws-{version}.jar	
lib\gluestd	aopalliance-1.0.jar	참고 Library
lib\gluestd	commons-codec-1.7.jar	
lib\gluestd	commons-dbcp-1.4.jar	
lib\gluestd	commons-lang3-3.1.jar	
lib\gluestd	commons-pool-1.6.jar	
lib\gluestd	ehcache-core-2.6.6.jar	
lib\gluestd	javaee-api-6.0.jar	
lib\gluestd	jaxp-api-1.4.2.jar	

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

lib\gluestd	jcl-over-slf4j-1.6.6.jar	
lib\gluestd	logback-classic-1.0.13.jar	
lib\gluestd	logback-core-1.0.13.jar	
lib\gluestd	slf4j-api-1.7.2.jar	
lib\gluestd	sqlite-jdbc-3.7.2.jar	
lib\gluestd	xmlbeans-2.4.0.jar	
lib\gluestd	spring-aop-3.2.0.RELEASE.jar	
lib\gluestd	spring-beans-3.2.0.RELEASE.jar	
lib\gluestd	spring-context-3.2.0.RELEASE.jar	
lib\gluestd	spring-context-support-3.2.0.RELEASE.jar	
lib\gluestd	spring-core-3.2.0.RELEASE.jar	
lib\gluestd	spring-expression-3.2.0.RELEASE.jar	
lib\gluestd	spring-jdbc-3.2.0.RELEASE.jar	
lib\gluestd	spring-orm-3.2.0.RELEASE.jar	
lib\gluestd	spring-tx-3.2.0.RELEASE.jar	
lib\gluestd	spring-web-3.2.0.RELEASE.jar	
lib\gluestd	spring-webmvc-3.2.0.RELEASE.jar	
lib\gluestd	spring-webmvc-portlet-3.2.0.RELEASE.jar	
templateFolder\java	build.xml	
templateFolder\java	build.properties	
templateFolder\java	model\sample.glue_uml_ad	
templateFolder\java	src\applicationContext.xml	
templateFolder\java	src\ehcache.xml	
templateFolder\java	src\logback-test.xml	
templateFolder\java	src\MyResource_en_US.properties	
templateFolder\java	src\MyResource_ko_KR.properties	
templateFolder\java	src\query\sample-query.glue_sql	
templateFolder\java	src\Run_GlueTester.java	
templateFolder\java	src\sample\activity\SearchEmp.java	
templateFolder\java	src\sample\vo\EmpVO.java	
templateFolder\java	src\service\sample-service.xml	
templateFolder\java	WebContents\springmvc\sample.jsp	
templateFolder\java	WebContents\springmvc\sample_global.jsp	
templateFolder\java	WebContents\springmvc\sample_vo.jsp	
templateFolder\java	WebContents\WEB-INF\dispatcher-servlet.xml	
templateFolder\java	WebContents\WEB-INF\web.xml	
templateFolder\maven	.classpath	Maven Type 의 Glue Project

Java Type 의 Glue Project
생성자원

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

templateFolder\maven	pom.xml	생성자원
templateFolder\maven	src\main\java\Run_GlueTester.java	
templateFolder\maven	src\main\java\sample\activity\SearchEmp.java	
templateFolder\maven	src\main\java\sample\vo\EmpVO.java	
templateFolder\maven	src\main\model\sample.glue_uml_ad	
templateFolder\maven	src\main\resources\applicationContext.xml	
templateFolder\maven	src\main\resources\ehcache.xml	
templateFolder\maven	src\main\resources\logback-test.xml	
templateFolder\maven	src\main\resources\MyResource_en_US.properties	
templateFolder\maven	src\main\resources\MyResource_ko_KR.properties	
templateFolder\maven	src\main\resources\query\sample-query.glue_sql	
templateFolder\maven	src\main\resources\service\sample-service.xml	
templateFolder\maven	src\main\webapp\springmvc\sample.jsp	
templateFolder\maven	src\main\webapp\springmvc\sample_vo.jsp	
templateFolder\maven	src\main\webapp\WEB-INF\dispatcher-servlet.xml	
templateFolder\maven	src\main\webapp\WEB-INF\web.xml	
templateFolder\maven	tomcat_setup.bat	

[표 1] GlueF/W 관련 라이브러리 정리

logback library 는 logback site(<http://logback.qos.ch>)에서 사용자가 직접 다운로드해서 GlueSDK 를 구성해야 한다.

3.2 WAS 환경설정

Glue Framework 기반의 web application 을 WAS 에 deploy 하기 위해서는 WAS(Web Application Server) 에 추가적인 환경 설정을 필요로 할 수 있다.

web application 에서 glue.properties 를 사용하고자 한다면, CONFIG_PATH 라는자바옵션을 WAS 에 설정해야 한다. CONFIG_PATH 는 Glue Framework 에서 예약한 시스템 Property 이다. glue.properties 가 WAS_HOME / usrconf 위치에 있다면, 다음과 같이 자바 옵션을 추가한다.

```
-DCONFIG_PATH=%WAS_HOME%/userconf
```

WAS 에 자바 옵션을 설정하는 방법은 해당 WAS 에서 제공하는 admin 가이드를 따른다.

glue.properties 는 key, value 로 구성된 텍스트 파일이며, 일부 예약어를 제외하고 사용자가 얼마든지 추가할 수 있다.

Glue Framework 에서 사용하는 Glue Property 의 예약어는 다음과 같다.

- audit.insert.key : dao 사용시 audit data 와 관련된 property
- audit.update.key : dao 사용시 audit data 와 관련된 property
- default.file.dir : 상대경로의 File 처리 activity 와 관련된 property
- default.msg.locale : 다국어 관련 property

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

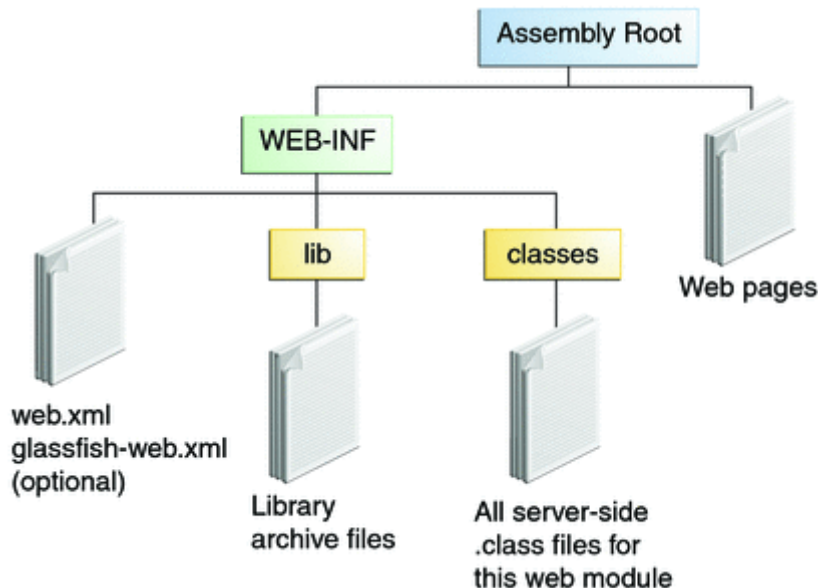
- default.ui.locale : 다국어 관련 property
- max.upload.file.size : File 처리 activity 와 관련된 property
- msg.parsing.type : ParseMessage, CreateMessage activity 와 관련된 property.

Glue Framework 에서 사용하는 시스템 Property(Java 옵션)는 다음과 같은 것을 사용하고 있다.

- CONFIG_PATH : glue.properties 파일 경로
- os.name : os 관련 java 예약어
- file.encoding : file 관련 java 예약어
- file.bigszie : file 관련 Glue 예약어
- cipher.path : 암호화와 파일 경로
- glue.scheduler.server.targeturl : scheduler 관련
- glue.scheduler.server.urldecoder.charset : scheduler 관련
- glue.scheduler.server.port : scheduler 관련
- glue.scheduler.server.address : scheduler 관련
- glue.scheduler.launcher : scheduler 관련

web application 실행시 loading 되어야 하는 library 위치는 SW 아키텍처 구성에 따른다.

web application 은 war 파일 또는 폴더의 형태로 설치(deploy)하며, 그 구조는 다음 그림과 같은 형태를 기본으로 한다.



[그림 11] web module structure

(그림출처 : <http://docs.oracle.com/javaee/6/tutorial/doc/>)

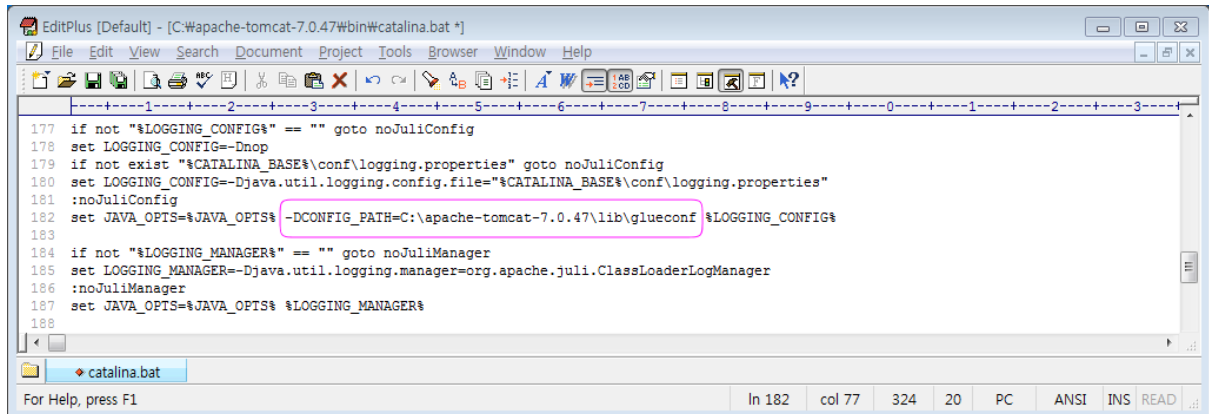
jar 와 같은 Library 를 web application 에 포함 시킬지 여부는 사용자의 선택이며, 포함시키지 않을 경우에는 추가적인 WAS 환경 설정을 필요로 한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

3.2.1 Tomcat 7

Tomcat 은 Java Servlet 과 JavaServer Page 를 구현한 오픈소스이며, 설치하는 Tomcat 사이트(<http://tomcat.apache.org>)를 참고 한다. 윈도우/리눅스 등 OS 별 설정 역시 Tomcat 를 참고 한다.

Core zip 을 다운받아 설치 했다면, glue.properties 파일의 위치는 bin/catalina.bat 파일에서 java option 인 CONFIG_PATH 를 통해 지정할 수 있다.



[그림 12] catalina.bat

web application 이 WEB-INF/lib 에 library 를 포함하지 않는다면, conf/catalina.properties 파일에 추가적으로 로딩할 library 또는 그 위치를 지정할 수 있다.

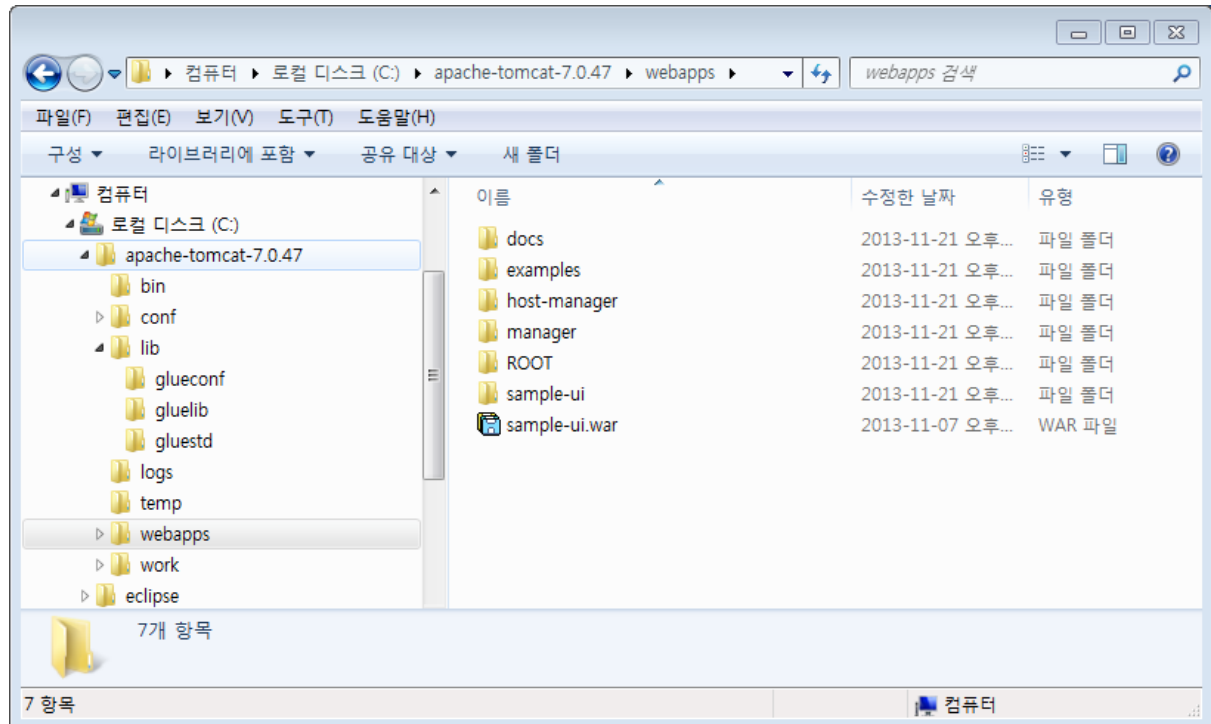
```
# 한줄에 common.loader 설정.
common.loader=${catalina.base}/lib,
${catalina.base}/lib/*.jar,
${catalina.home}/lib,
${catalina.home}/lib/*.jar,
${catalina.home}/usr/lib/gluelib,
${catalina.home}/usr/lib/gluelib/*.jar,
${catalina.home}/usr/lib/gluestd,
${catalina.home}/usr/lib/gluestd/*.jar
```

Glue 라이브러리 설정후 추가적으로 cahce.logging 관련 설정을 하여야 한다. 사용하는 Cache 와 Logging library 에 따라 그 설정 파일과 환경 구성 방법은 여기서 다루지 않는다.

Cache 설정은 conf/catalina.properties 에 설정한 class path 를 참고해서 cache.ccf(JCS) 또는 ehcache.xml(Ehcache) 파일을 둔다. cache 설정파일은 gluestd 에 속하는 library 와 같은 위치에 있어야 한다.

다음은 tomcat 설치 모습이다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

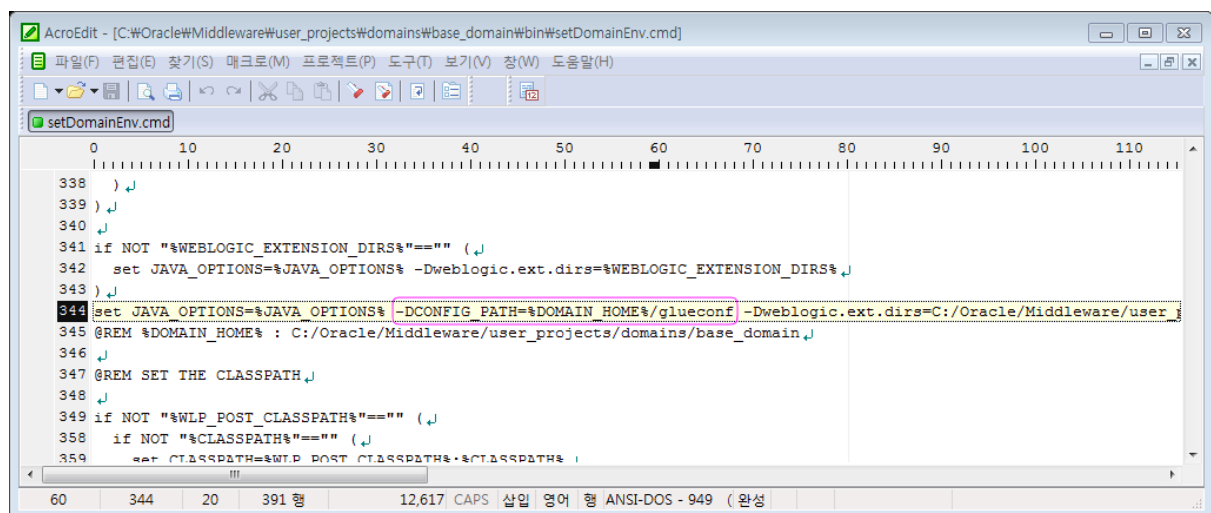


[그림 13] Tomcat

3.2.2 Weblogic 12C

WebLogic Server 설치는 Oracle Site(<http://www.oracle.com>)를 참고 한다.

Oracle WebLogic Server 12c 를 설치해서 domain 까지 생성했다면, glue.properties 파일의 위치는 bin/ setDomainEnv.cmd 파일에서 java option 을 통해 지정할 수 있다.



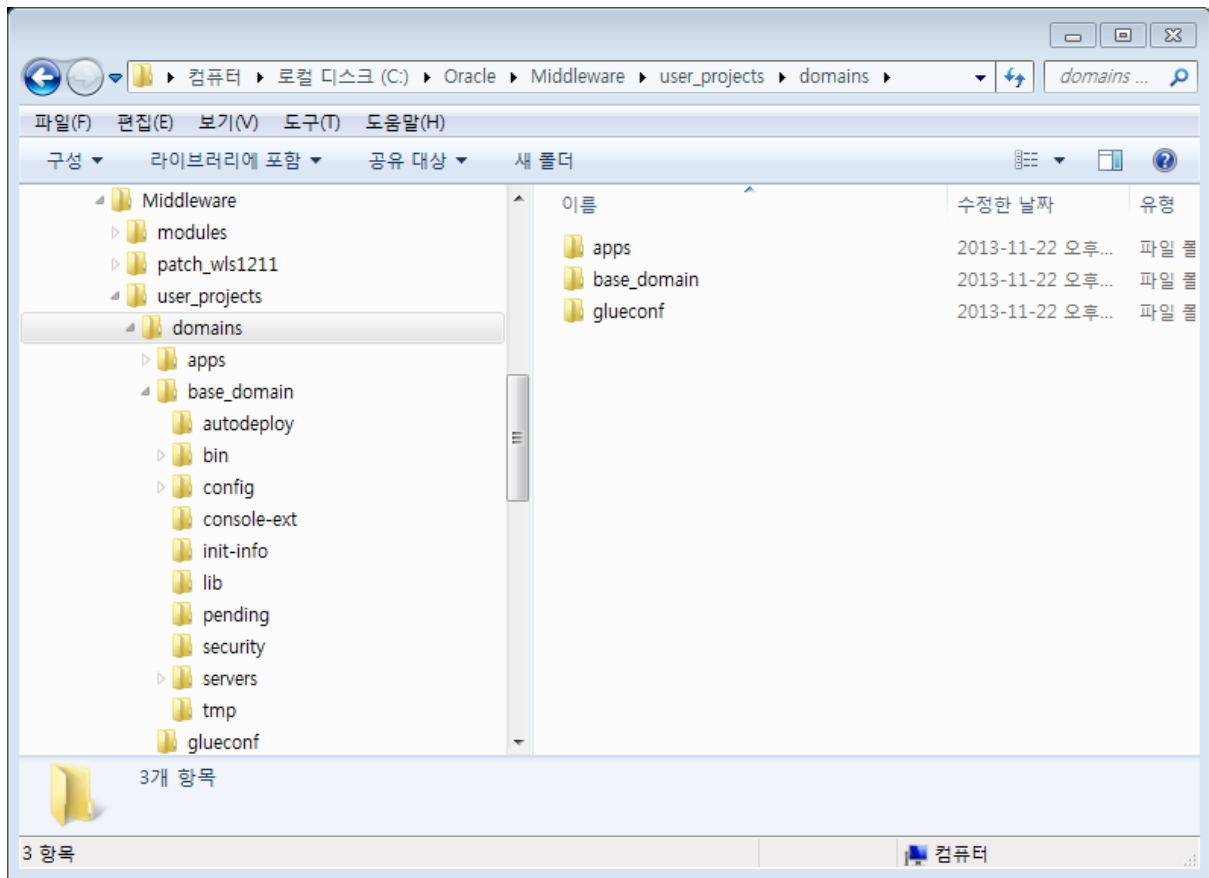
[그림 14] setDomainEnv.cmd

WebLogic 에서는 web application 이 WEB-INF/lib 에 library 를 포함하지 않는다면, weblogic.ext.dirs 이라는 java 옵션을 통해 추가적으로 로딩할 library 또는 그 위치를 지정할 수 있다.

-Dweblogic.ext.dirs=C:/Oracle/Middleware/user_projects/domains/apps

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

다음은 weblogic 설치 모습이다.

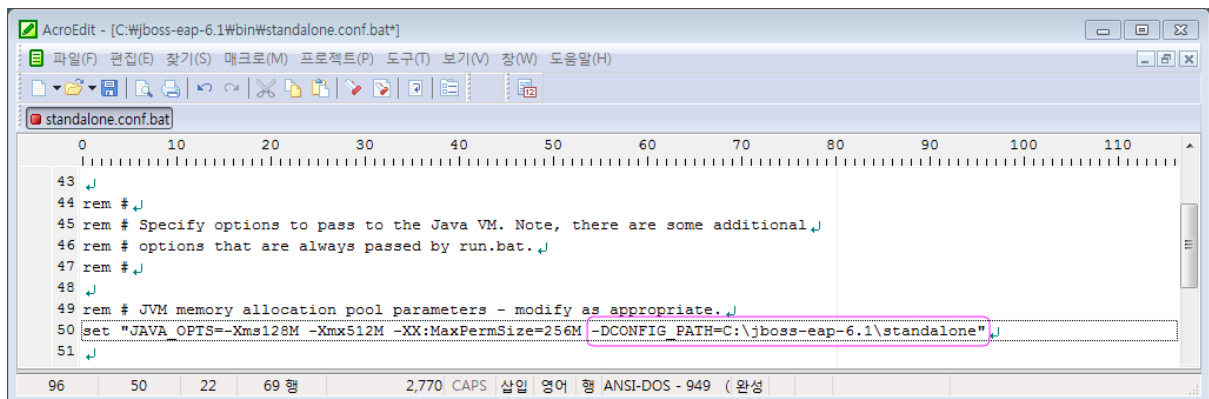


[그림 15] WebLogic

3.2.3 JBoss EAP 6

JBoss 설치 는 JBoss Site([http:// www.jboss.org](http://www.jboss.org))를 참고한다.

JBoss EAP 6.1.0 을 설치 했다면, Standadalone Server 모드와 Managed Domain 모드를 선택해서 JBoss EAP 6 를 시작 할 수 있다. Standalone Server 로 시작한다고 하면, glue.properties 파일의 위치를 bin/standalone.conf.bat 에 Java Option 을 통해 지정한다.

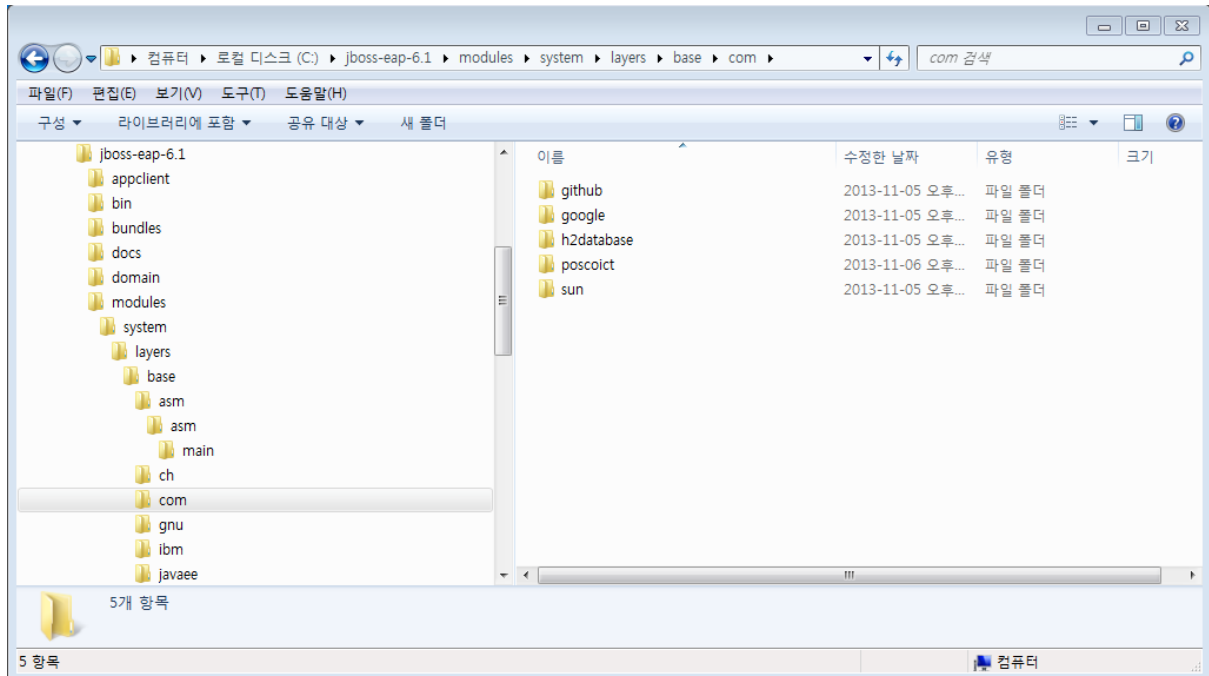


[그림 16] standalone.conf.bat

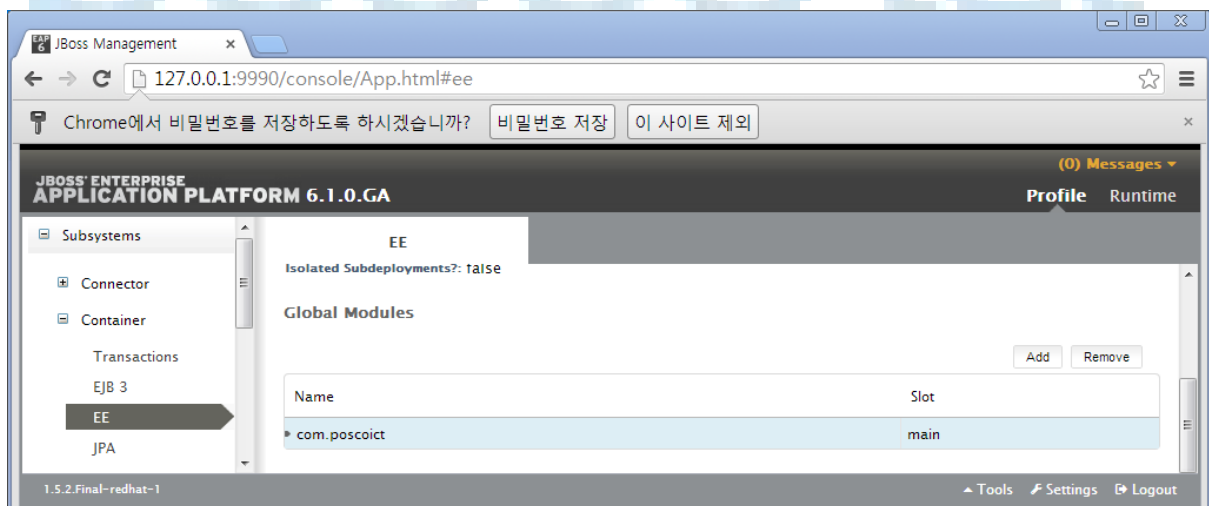
	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

web application 이 WEB-INF/lib 에 library 를 포함하지 않는다면, module 을 구성해야 한다. JBoss 에서는 클래스 로딩을 위한 classes 의 논리적 그룹을 모듈이라 하고, static / dynamic 이라고 불리는 2 가지 형태로 구분한다. Module 구성에 관한 것은 JBoss 사이트의 administration guide 를 참고한다.

modules/system/layers/base 위치에 하위 directory 를 만들어 jar 파일과 module.xml 을 위치 시키고, JBoss Console 화면을 통해 Profile -> Container -> EE 부분에서 Global Module 을 추가할 수 있다.



[그림 17] JBoss



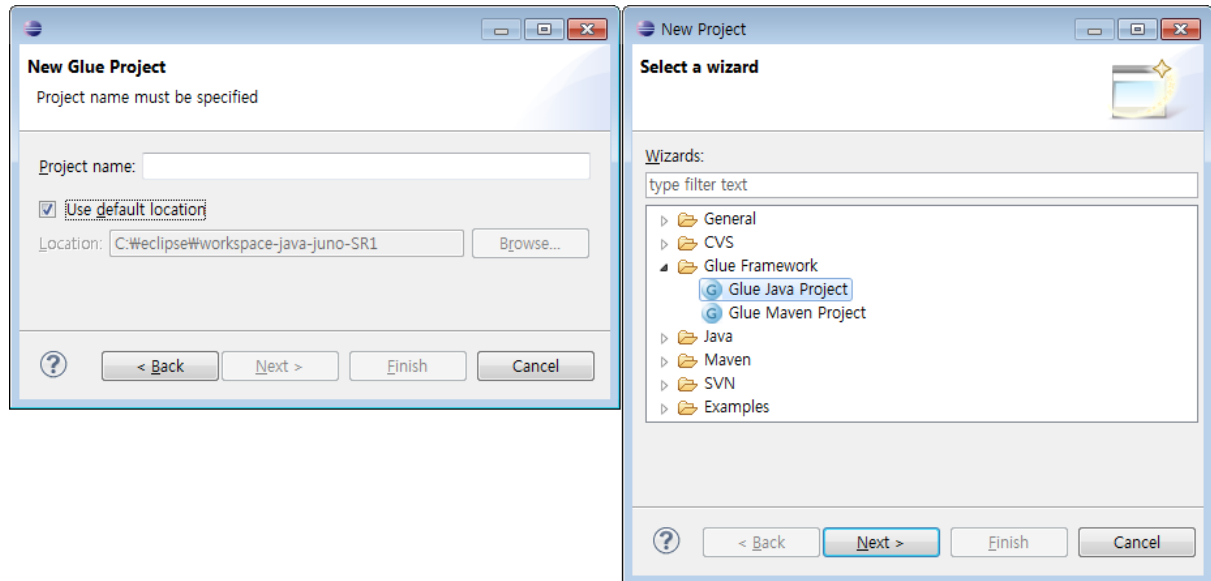
[그림 18] JBoss Console

3.3 Glue Project

Eclipse IDE 의 File 메뉴의 New -> Project 에서 Glue Java Project 또는 Glue Maven Project 를 선택해서 Glue Project 를 생성한다. 2 가지 유형의 프로젝트는 GlueSDK 의 templateFolder 에 있는 파일들을 이용한다.

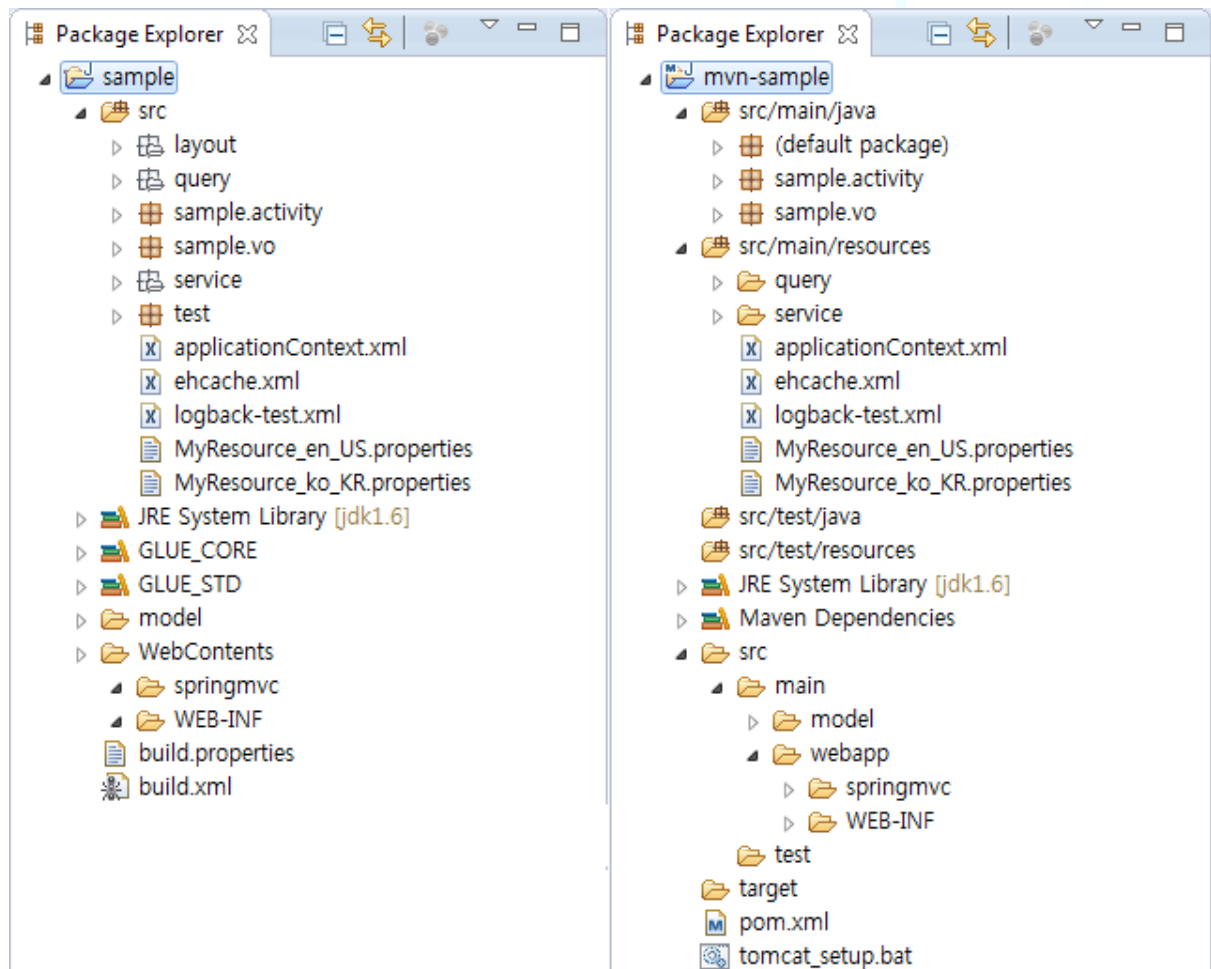
	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Glue Java Project 형태로 생성하면 ant 를 이용할 수 있는 형태로 프로젝트가 생성된다. Glue Maven Project 형태로 생성하면 mvn 을 이용할 수 있는 형태로 프로젝트가 생성된다.



[그림 19] Glue Project 생성 wizard

다음은 Glue Java Project 와 Glue Maven Project 로 프로젝트를 생성한 결과이다.



[그림 20] Glue Project 구조

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

3.3.1 Glue Java Project

Glue Java Project 형태에서는 GLUE_CORE 와 GLUE_STD 라는 user library 자동으로 설정된다. GlueSDK 의 gluelib 와 gluestd 에 있는 라이브러리들로 자동 구성된다. GlueSDK 의 gluelib 와 gluestd 의 라이브러리가 변경될 경우, Glue SDK Location 을 설정하는 Preference 창 의 apply 버튼을 이용해 user library 를 재로딩 할 수 있다.

GlueSDK 의 gluestd 에는 최소한의 라이브러리로 구성되어 있으며, 추가적인 기능을 사용하고자 한다면, gluestd 에 해당 라이브러리를 추가해야 한다. 일부 오픈소스 라이브러리는 gluestd_ 로 시작하는 폴더에 있으며, 빠진 라이브러리들은 해당 라이브러리를 제공하는 사이트에서 다운받아 구성하여야 한다.

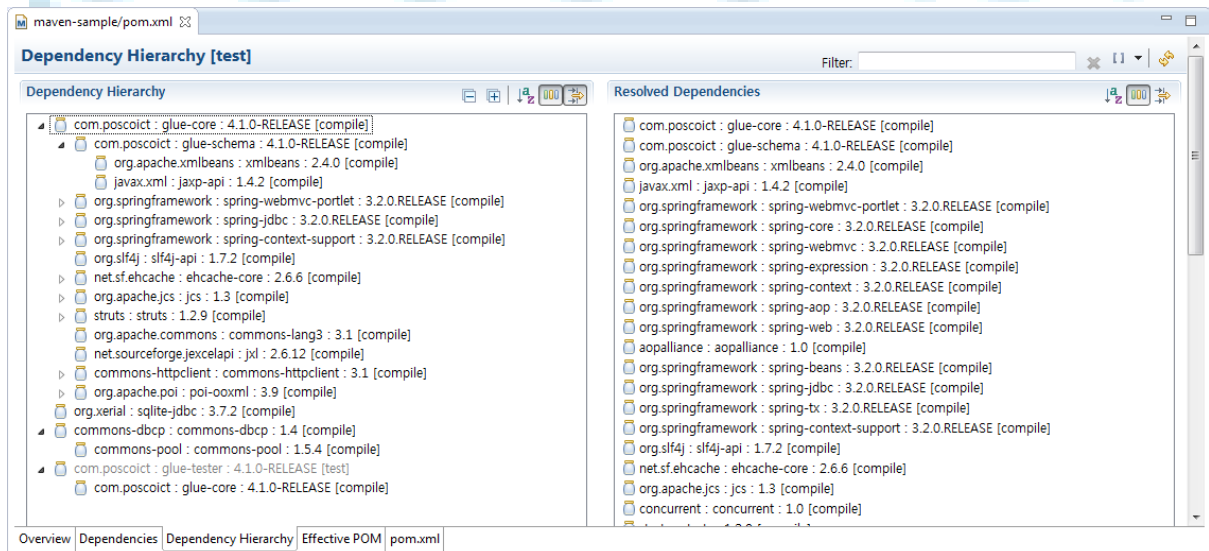
3.3.2 Glue Maven Proejct

Glue Maven Project 에서는 Maven Dependencies 를 통해 프로젝트에서 사용하는 Library 가 구성된다. pom.xml 파일에 다음과 같은 dependency 를 추가하기 위해서는 따로 Repository 를 구성하여야 하며, Repository 의 구성은 여기서는 다루지 않는다.

아래와 같은 형태에서 artifactId 와 version 은 GlueSDK / gluelib 에 있는 jar 파일의 이름으로 참고해서 구성하도록 한다.

```
<dependency>
  <groupId>com.poscoict</groupId>
  <artifactId>glue-core</artifactId>
  <version>4.1.0-RELEASE</version>
</dependency>
```

Eclipse IDE 에서의 glue-core 의 의존도는 아래와 같이 확인할 수 있다.



[그림 21] pom.xml 의 Dependency Hierachy

pom.xml 에서 불필요하다고 판단되는 libaray 는 다음과 같이 제외시킬 수 있다. 다음은 JCS 와 Struts 를 제외시킨 모습이다.

```
<dependency>
  <groupId>com.poscoict</groupId>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

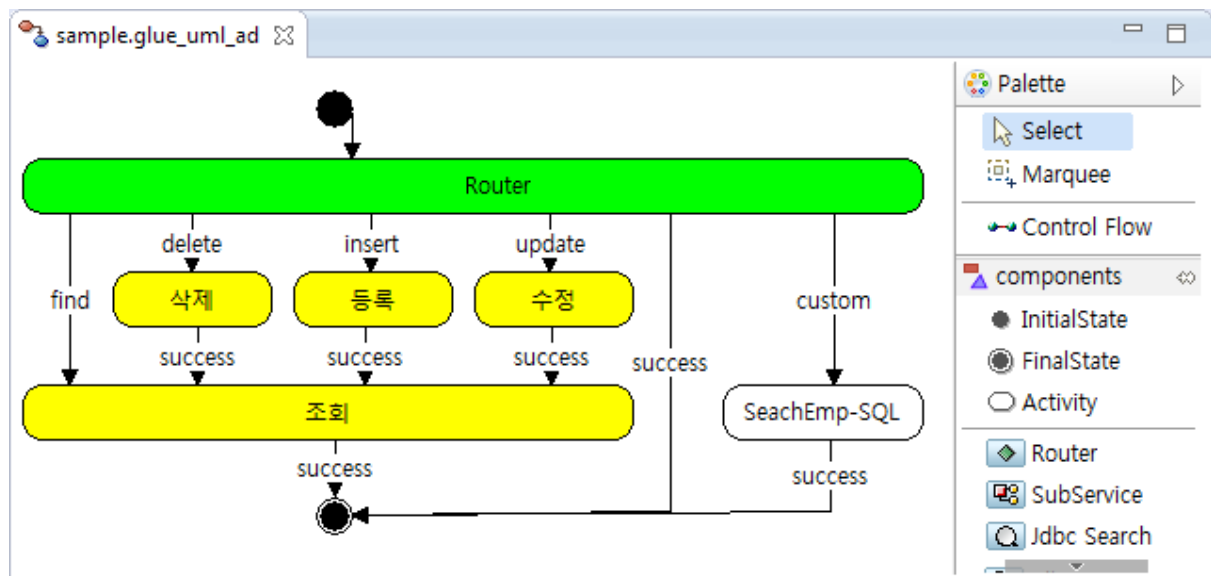
```

<artifactId>glue-core</artifactId>
<version>4.1.0-RELEASE</version>
<exclusions>
  <exclusion>
    <artifactId>jcs</artifactId>
    <groupId>org.apache.jcs</groupId>
  </exclusion>
  <exclusion>
    <artifactId>struts</artifactId>
    <groupId>struts</groupId>
  </exclusion>
</exclusions>
</dependency>

```

3.4 Glue Activity Diagram

Activity Diagram 은 단위 Business Flow 그림으로 아래와 같이 표현한 것이다.

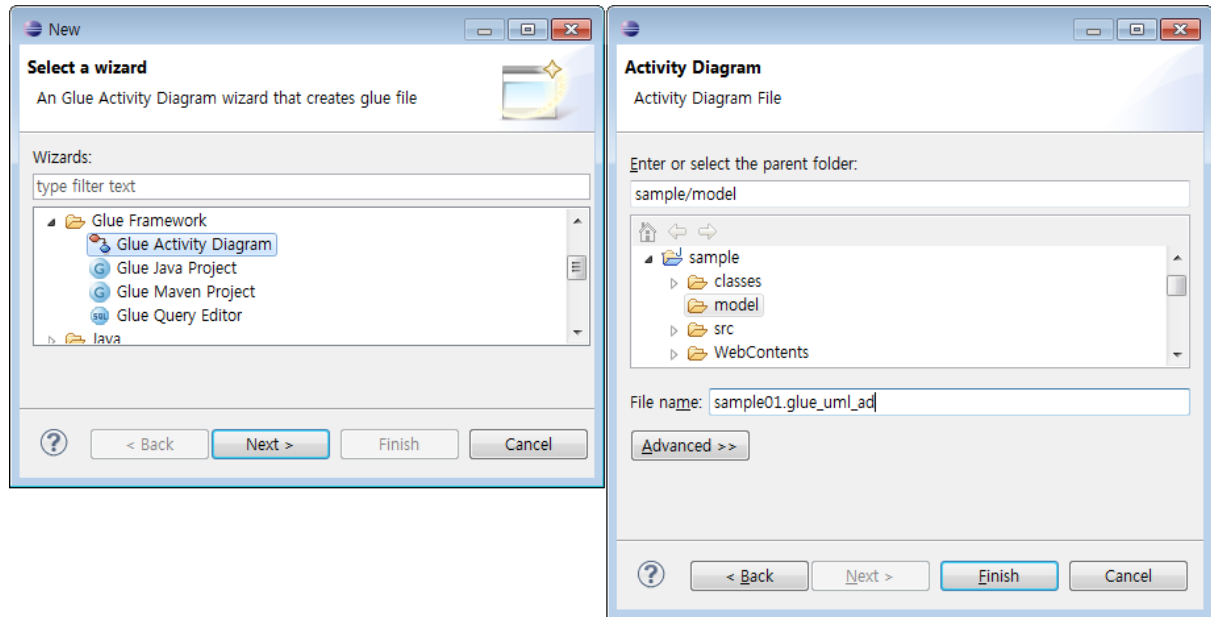


[그림 22] Activity Diagram

Glue Activity Diagram 의 Palette 의 Control Flow 와 Componets 를 이용해 Business Flow 를 설계한다. Components 영역에서 상단의 3 개(Initial State 와 Final State 와 Activity)이후에 보이는 것은 Reuse Activity 이다. Reuse Activity 는 GlueSDK 에 포함된 glue-config.xml 를 통해 추가/삭제 할 수 있다.

Eclipse IDE 의 File 메뉴에서 New -> other 에서 Glue Activity Diagram 을 선택해서 Activity Diagram 을 생성할 수 있다. 확장자는 glue_uuml_ad 이다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 23] Glue Activity Diagram 생성 wizard

다음은 파일 생성 wizard 를 이용해 Glue Activity Diagram 을 생성한 모습입니다. 캔버스에 Palette 의 components 를 이용하면 된다.



[그림 24] Glue Activity Diagram

다음은 GlueSDK 에 포함된 glue-config.xml 에서 activity 를 정의한 부분이다. 전체 glue-config.xml 은 GlueSDK/config/glue-config.xml 을 보도록 한다. Palette 에 보이는 Reuse Activity 의 순서는 glue-config.xml 에 작성된 순서와 같으며, Reuse Activity 를 Grouping 해서 보이게 하고자 한다면, addSeperator 속성을 이용하면 된다. Reuse Activity 를 색상 변경하고자 한다면, color 요소 값을 변경하면 된다.

```
<activity name="Router" icon="icons/activity_router.gif" addSeperator="true">
  <bubble-tip>Default Router Activity</bubble-tip>
  <template>Glue Router template</template>
  <color>green</color>
  <default-properties>
    <property name="class"
      value="com.poscoict.glueframework.biz.activity.GlueDefaultRouter" />
  </default-properties>
</activity>
```

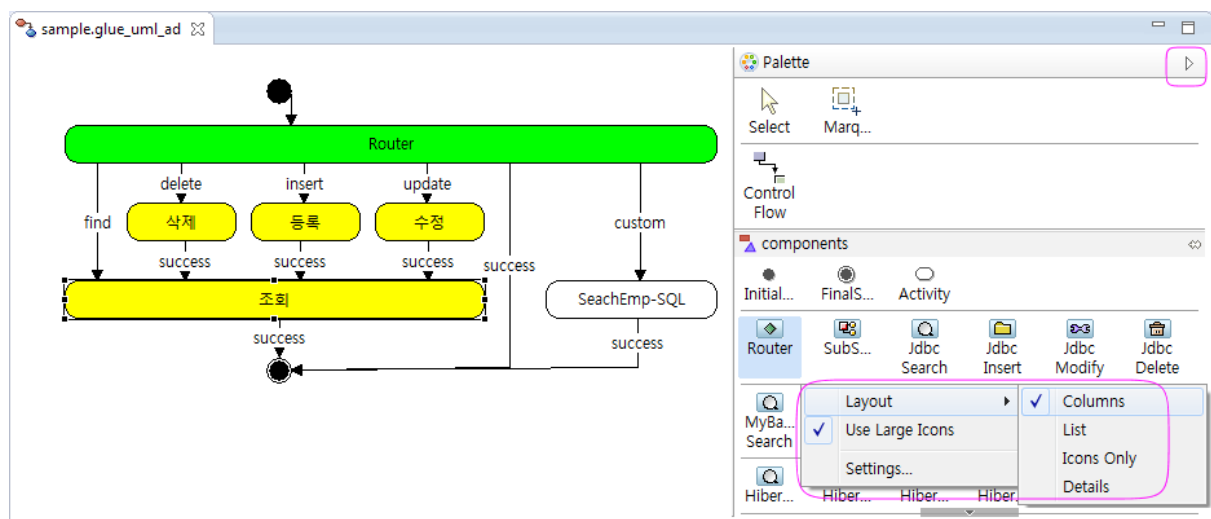
	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<activity name="SubService" icon="icons/activity_subservice.gif">
  <bubble-tip>SubService Activity</bubble-tip>
  <template>Glue SubService template</template>
  <color>lightBlue</color>
  <default-properties>
    <property name="class"
      value="com.poscoict.glueframework.biz.activity.GlueSubService" />
    <property name="service-name" value="sample-service" />
    <property name="new-transaction" value="false" />
  </default-properties>
  ...

```

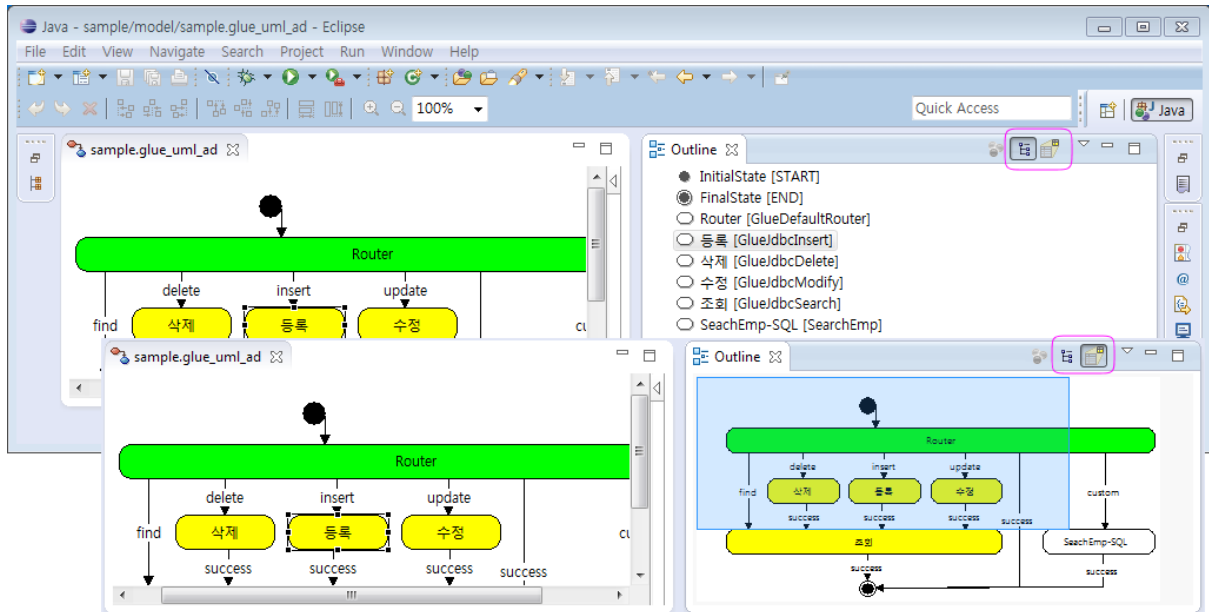
Activity Diagram 작성시 Palette의 내용은 상단의 삼각형 모양을 클릭해보면 된다. Palette가 펼쳐지면, Palette의 Layout은 다음과 같이 Columns, List, Icons Only, Details을 선택할 수 있다.



[그림 25] Palette의 Layout

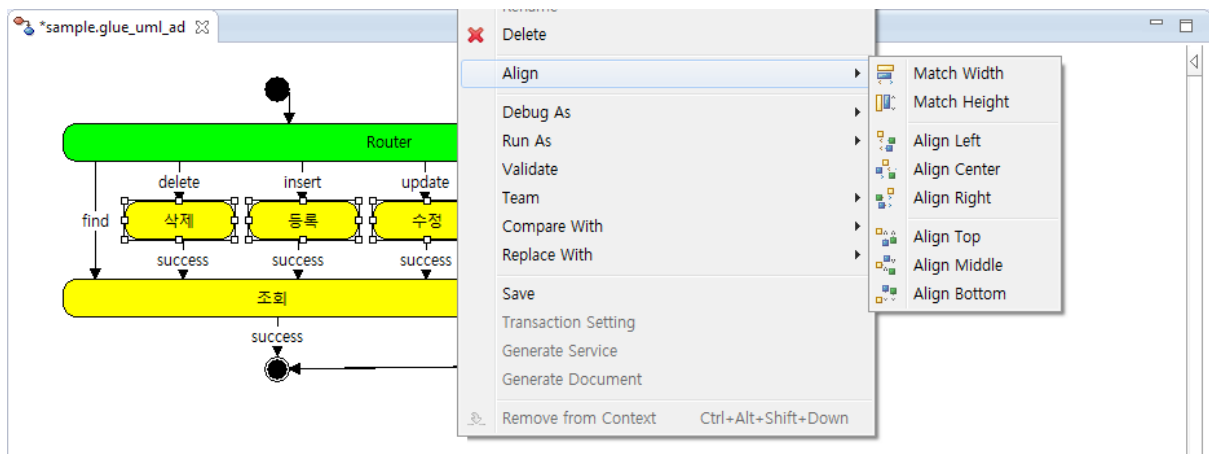
다음과 같이 Outline 뷰를 통해서 Activity Diagram의 구성요소를 확인 할 수 있으며, 구성 요소를 클릭하면 해당 Activity로 이동 할 수 있다. 그리고 Outline 뷰의 상단 아이콘을 통해 Activity 목록을 보거나 축소된 모습을 볼 수 있다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 26] Activity Diagram 의 Outline 뷰

Activity 의정렬 기능은 다음과 같이 이용할 수 있다. 정렬 하고자 하는 Activity 를 선택한 후, 마우스 오른쪽 버튼의 팝업 메뉴를 통해 좌우정렬, 상하 정렬 선택하면 된다.



[그림 27] Activity 정렬

3.4.1 Activity

Activity Diagram 의 구성 요소인 Activity 는 name, class, transition, property 로 이루어져 있다. transition 과 property 는 name 과 value 속성을 갖는다.

```
<activity name="로직 1" class="sample.activity.BizLogic">
  <transition name="continue" value="NextActivityName"/>
  <transition name="success" value="end"/>
  <property name="condition1" value="static_data_1"/>
  <property name="condition2" value="static_data_2"/>
</activity>
```

name 은 해당 Activity Diagram 에서 유일한 이름이어야 한다.

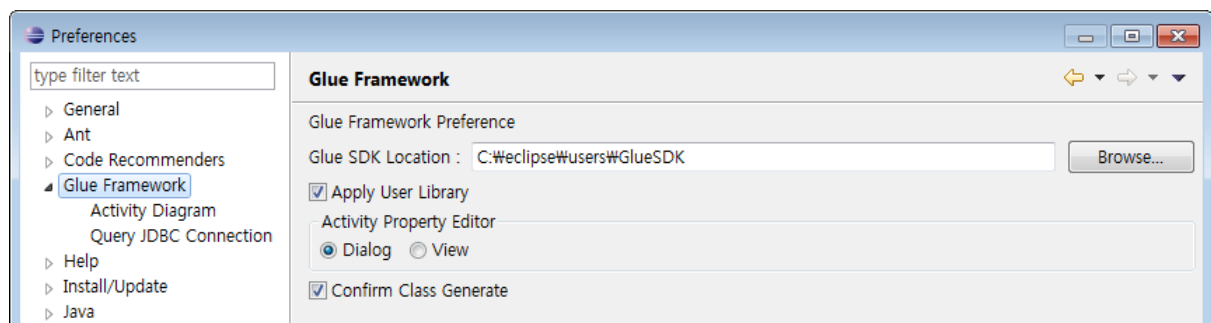
	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

class 는 GlueActivity 를 상속받은 Class 이어야 한다. Reuse Activity 가 아닌 경우 GlueActivity 를 상속받아 구현해야한다.

transition 은 name 과 value 속성으로 구성된다. 현재 Activity 에서 transition 의 name 을 중복 되지 않게 한다. transition 의 value 는 다음 Activity 의 name 을 나타낸다. Activity 는 반드시 하나 이상의 transition 을 갖는다. Reuse Activity 일 경우 반드시 "success" transition 을 가지고 있어야 한다. transition 의 value 가 "end" 이면, Final State 로 Control Flow 가 작성되어 있음을 의미한다.

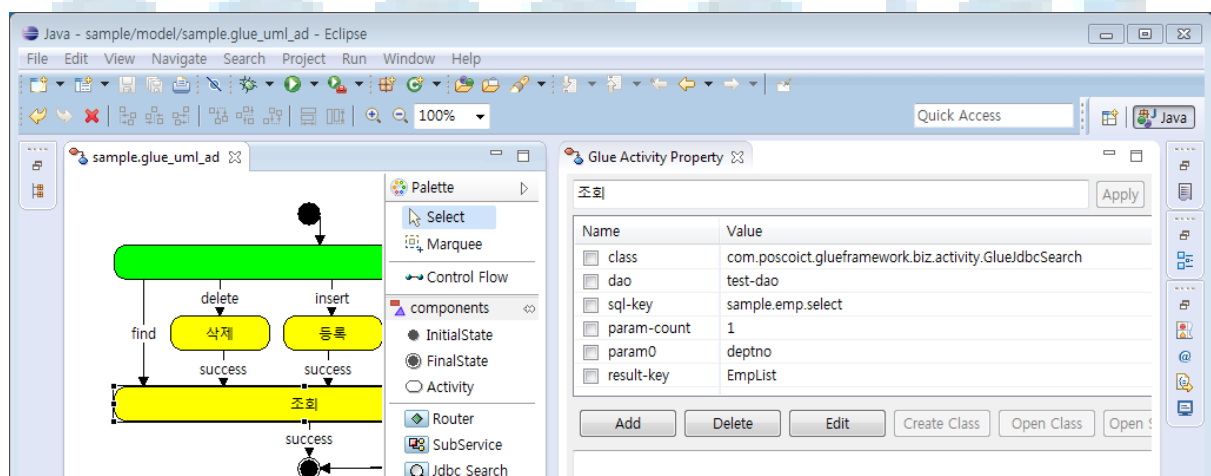
property 는 name 과 value 속성으로 구성되며, 해당 Activity 에서만 사용할 수 있는 사용자 정의 Data 이다. Activity 에서 property 의 개수 제한은 없으나, property 의 name 을 중복되지 않게 한다.

Activity Diagram 에서 Activity 의 Property 를 관리하는 화면은 View 와 Diaglog 를 제공한다. 다음과 같은 Preference 에서 Dialog 을 사용할 것인지 View 을 사용할 것인지 선택할 수 있다.



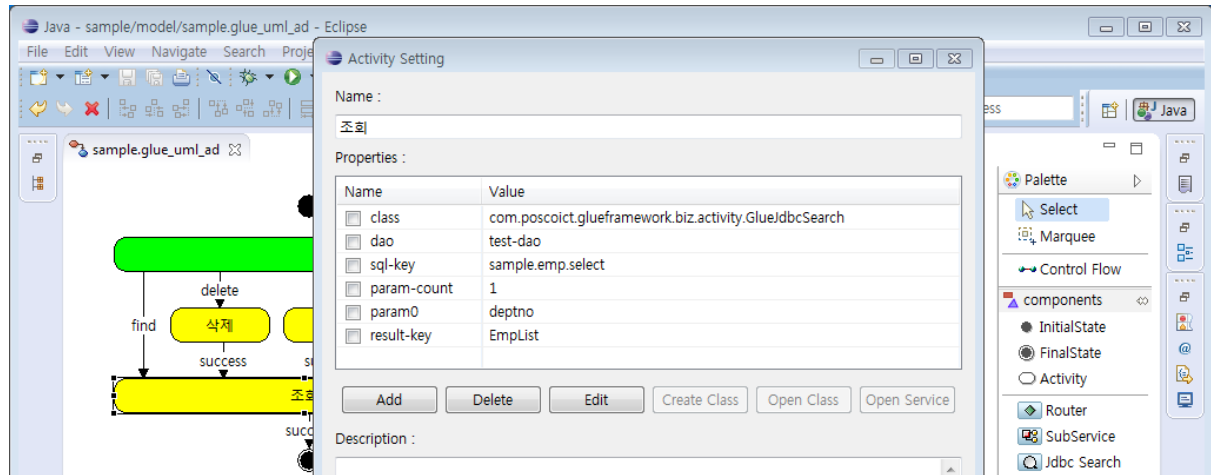
[그림 28] Activity Property Editor

다음은 View 형태와 Dialog 형태의 Activity Property Editor 이다.



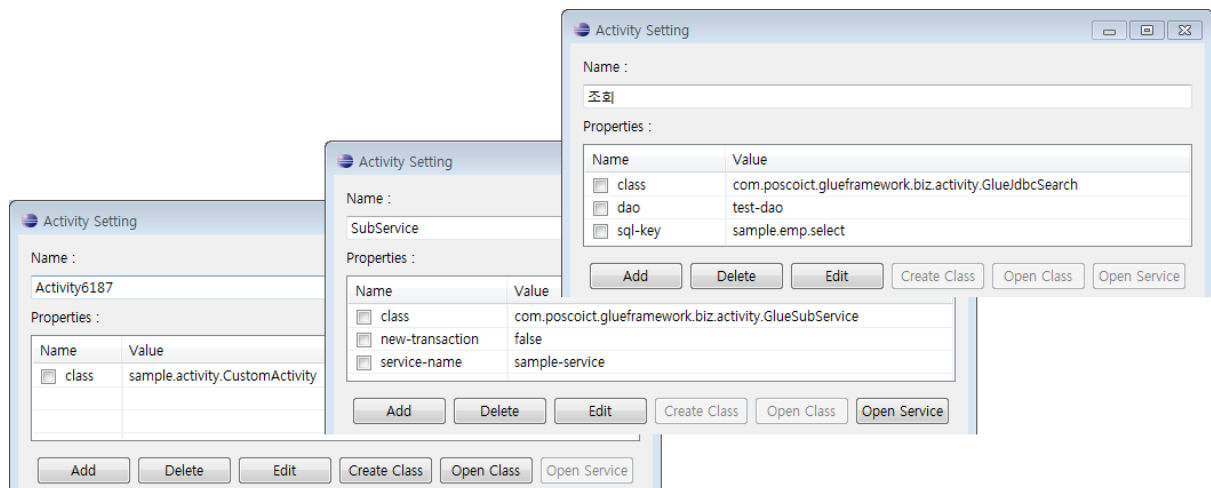
[그림 29] View 형태의 Activity Property Editor

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 30] Dialog 형태의 Activity Property Editor

Activity의 유형에 따라 아래와 같이 Activity Setting 창의 버튼 활성화가 달라진다.



[그림 31] Activity Setting의 버튼

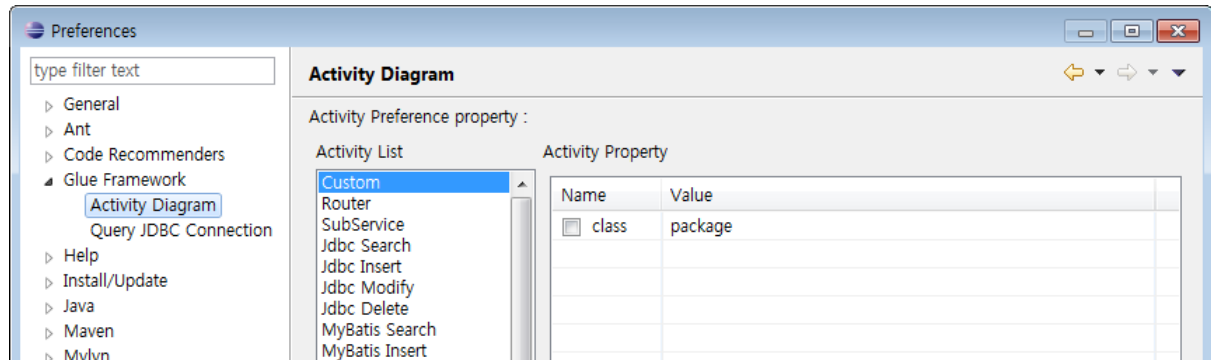
Reuse Activity의 경우 property를 편집을 위한 Add/Delete/Edit 버튼이 활성화 되면, Reuse Activity의 경우 SubService 일 경우 Open Service가 추가로 활성화 된다. Custom Activity의 경우 Create Class/Open Class 버튼이 활성화 된다. GlueSDK/config/glue-config.xml의 class generator 부분은 Create Class와 Open Class 버튼의 활성화여부와 관계된다.

Activity의 Property의 name 부분에는 check box가 있으며, 이를 이용해 삭제하고자 하는 property를 선택해서 삭제할 수 있다. 여러 개의 Property를 수정하고자 한다면, Edit버튼을 활용한다.

3.4.2 Reuse Activity

Glue Activity Diagram 작성시 Reuse Activity를 Palette에서 확인 할 수 있으며, Reuse Activity의 Property는 Preference 창에서 확인 할 수 있다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



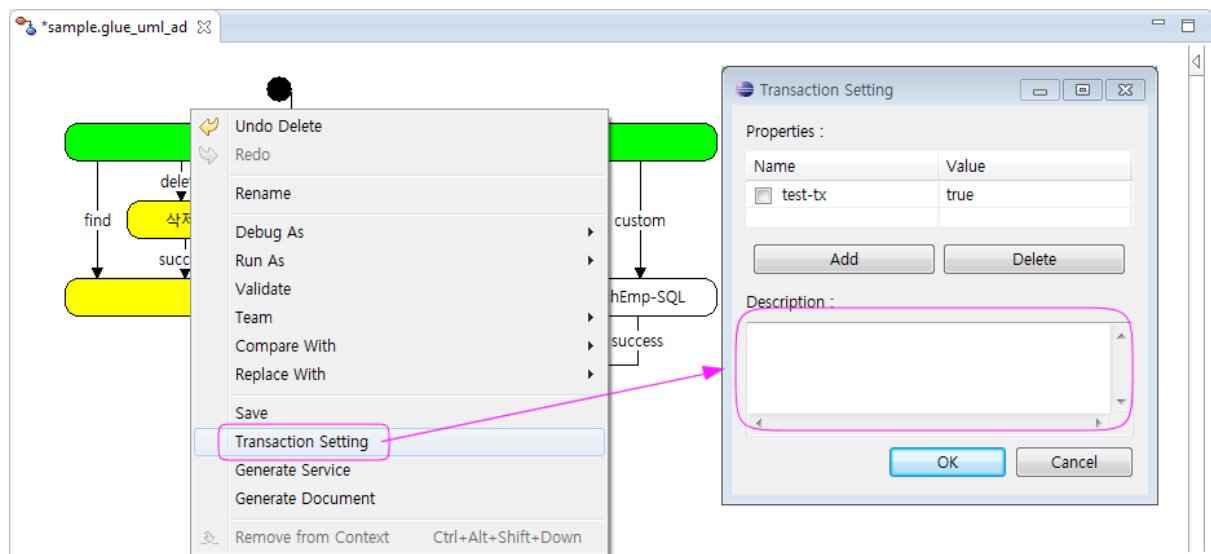
[그림 32] Reuse Activity List

Reuse Activity 의 Property 는 GlueSDK / config / glue-config.xml 에서 수정할 수 있으며, Preference 창에서는 해당 Reuse Activity 의 Property 를 확인하는 기능을 제공하고 있다.

3.4.3 서비스명세서

Activity Diagram 작성시, Activity Diagram 단위로, Activity 단위로 Description 을 작성 할 수 있다.

Activity Diagram 에 대한 desription 은 그림과 같이 Transaction Setting 창을 통해 작성하고,



[그림 33] Activity Diagram 의 Description

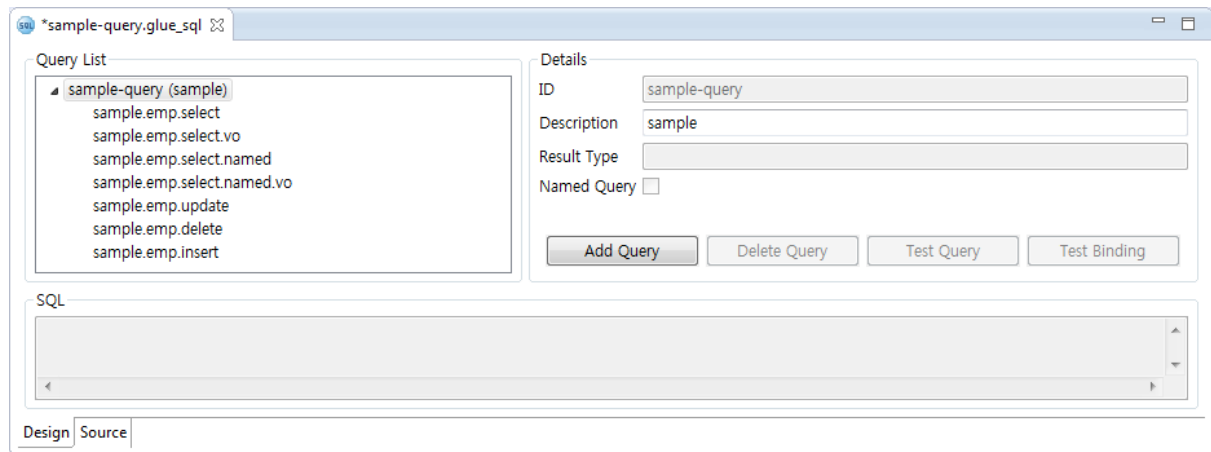
Activity 에 대한 description 은 Property Setting 창을 통해 작성한다.

이렇게 작성된 Description 를 포함해서 서비스 명세서를 생성할 수 있다. 서비스 명세서는 Activity Diagram 의 팝업 메뉴에서 Generate Document 를 이용한다.

3.5 Glue Query Editor

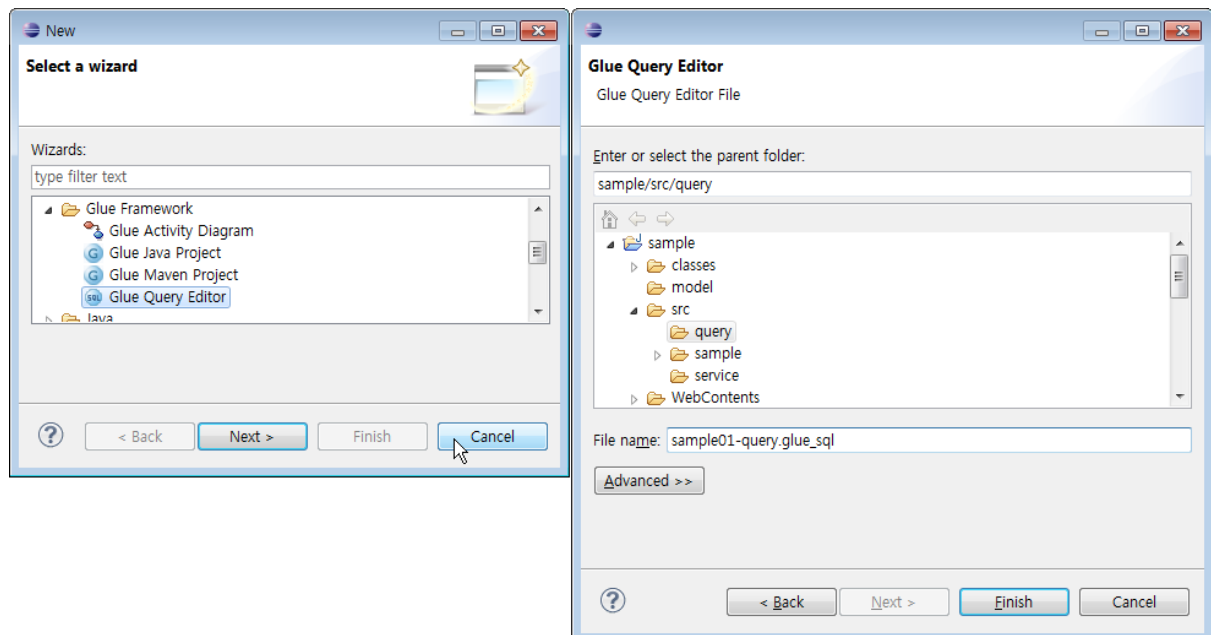
Glue Query Editor 을 통해 SQL 을 파일로 따로 관리 할 수 있다. 다음 그림과 같이 SQL 별로 query id 를 부여해서 query id 별로 SQL 을 관리하는 것이다. Query List 영역에 id 를 선택하면 SQL 영역에서 그 내용을 확인 할 수 있다. 하단의 Source Tab 으로 이동하면 해당 파일에 등록된 전체 SQL 문들을 확인 할 수 있다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



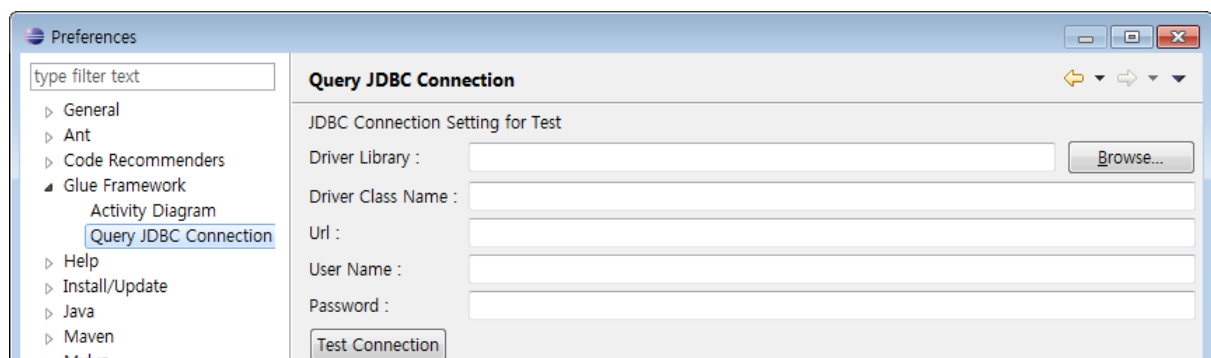
[그림 34] Glue Query Editor

Eclipse IDE 의 File 메뉴에서 New -> other 에서 Glue Query Editor 를 선택해서 Query 파일을 생성할 수 있다. 해당 프로젝트의 src/query 위치에 생성하며, 확장자는 glue_sql 이다



[그림 35] Glue Query Editor 생성 wizard

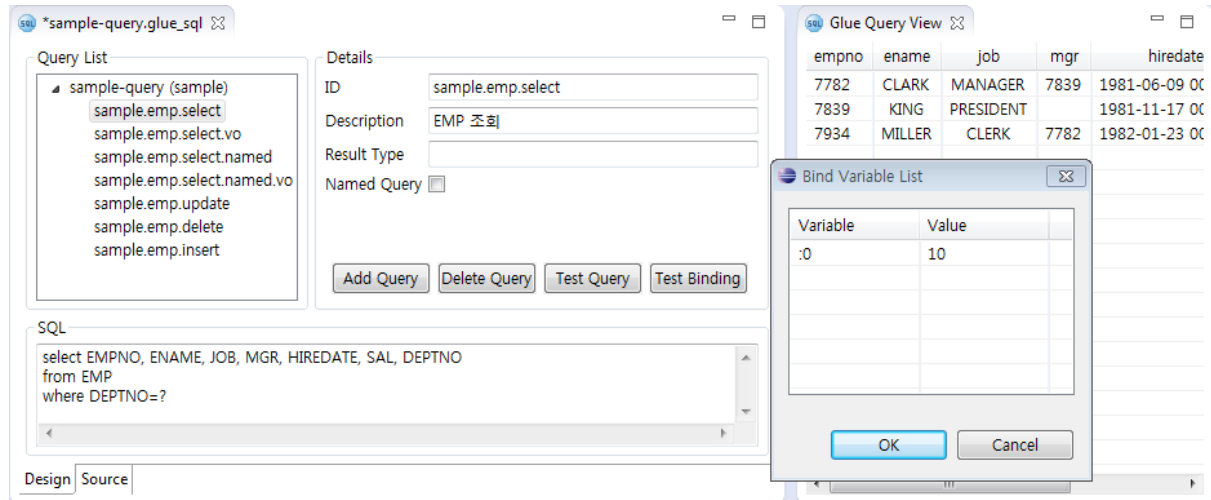
다음과 같이 JDBC Connection 정보와 JDBC Driver 를 지정하면, 조회용 SQL 문의 경우 테스트가 가능하다.



[그림 36] Query JDBC Connection

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

SQL 은 2 가지 Type 으로 구분되며, Named Query 가 아니면서 조회용인 SQL 을 테스트할 수 있다. 즉 Glue Query Editor 의 Detail 영역의 Test Query 버튼이 활성화 되는 경우이다.



[그림 37] Test Query

posco ict

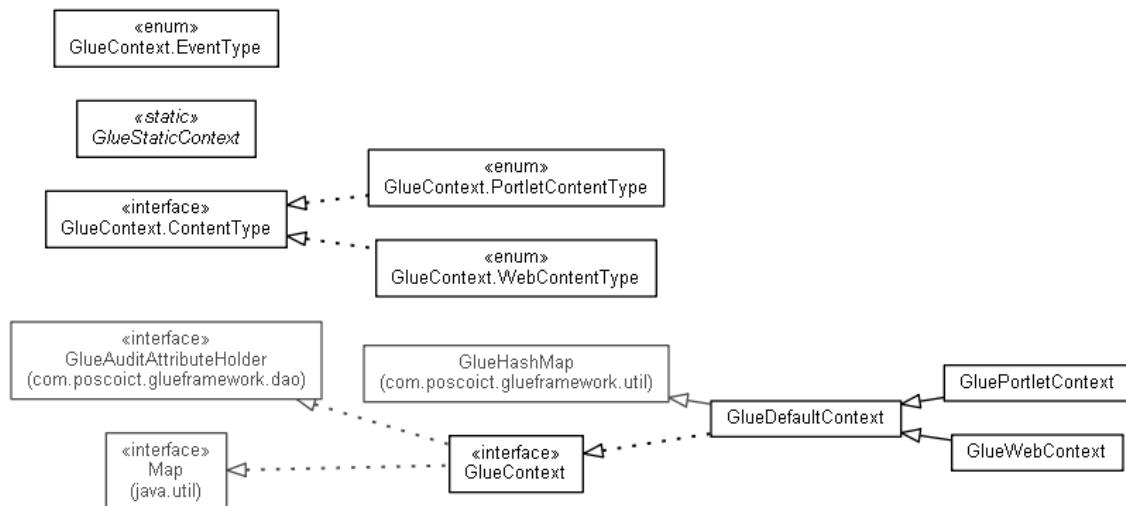
	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

4. GlueContext

GlueContext 는 내부저장소 interface 이다. GlueContext 는 GlueService 단위로 생성되며, Event 의 유형에 따라 3 가지 중 하나로 생성된다. 내부저장소는 Map 형태로 Data 를 관리하며, Event 의 유형에 구성되는 Data 의 성격이 다르다. Event 의 유형은 웹, 포틀릿, 그외의 것으로 구분한다.

- GlueDefaultContext : 웹과 포틀릿을 제외한 Event 일 경우로, Root Map 와 Message List 를 갖는다.
- GlueWebContext : 웹 Event 일 경우로 , HttpRequest(ServletRequest)의 Parameter 와 Attribute 와 HttpSession, Cookie, Header Data 를 별도의 Map 형태로 포함하며, MultipartRequest 를 갖는다.
- GluePortletContext : 포틀릿 Event 일 경우로, PortletRequest 의 Public Parameter, Private Parameter, Attribute 와 PortletSession, ApplicationSession, Cookie, Header Data 를 별도의 Map 으로 갖는다.

다음은 GlueContext 와 관련된 Class 관계도 이다.



[그림 38] GlueContext 의 Class 관계

열거형의 EventType 에는 DEFAULT, WEB_BROWSER, EAI_SERVER, EDGE_SERVER 가 있다. EventType 은 다음과 같이 GlueContext 생성시사용된다.

```

GlueContext ctx1 = new GlueDefaultContext(); // EventType.DEFAULT 임
GlueContext ctx2 = new GlueDefaultContext(EventType.EAI_SERVER);
GlueContext ctx3 = new GlueWebContext();//EventType.WEB_BROWSER 임
GlueContext ctx4 = new GluePortletContext();//EventType.WEB_BROWSER 임
  
```

ContentType 은 GlueWebContext 와 GluePortletContext 의 Root Map 을 제외한 추가적인 Map 의 유형을 나타낸다. GlueWebContext 에서는 HTTP_REQUEST_PARAM, HTTP_REQUEST_ATTR, SESSION_ATTR, COOKIE, HTTP_HEADER, MULTIPART 라는 유형이 있고, GluePortletContext 에는 PORTLET_REQUEST_PARAM_PUBLIC, PORTLET_REQUEST_PARAM_PRIVATE, PORTLET_REQUEST_ATTR, SESSION_ATTR_APPLICATION_SCOPE, SESSION_ATTR_PORTLET_SCOPE, COOKIE, HTTP_HEADER 유형이 있어서 다음과 같이 사용할 수 있다.

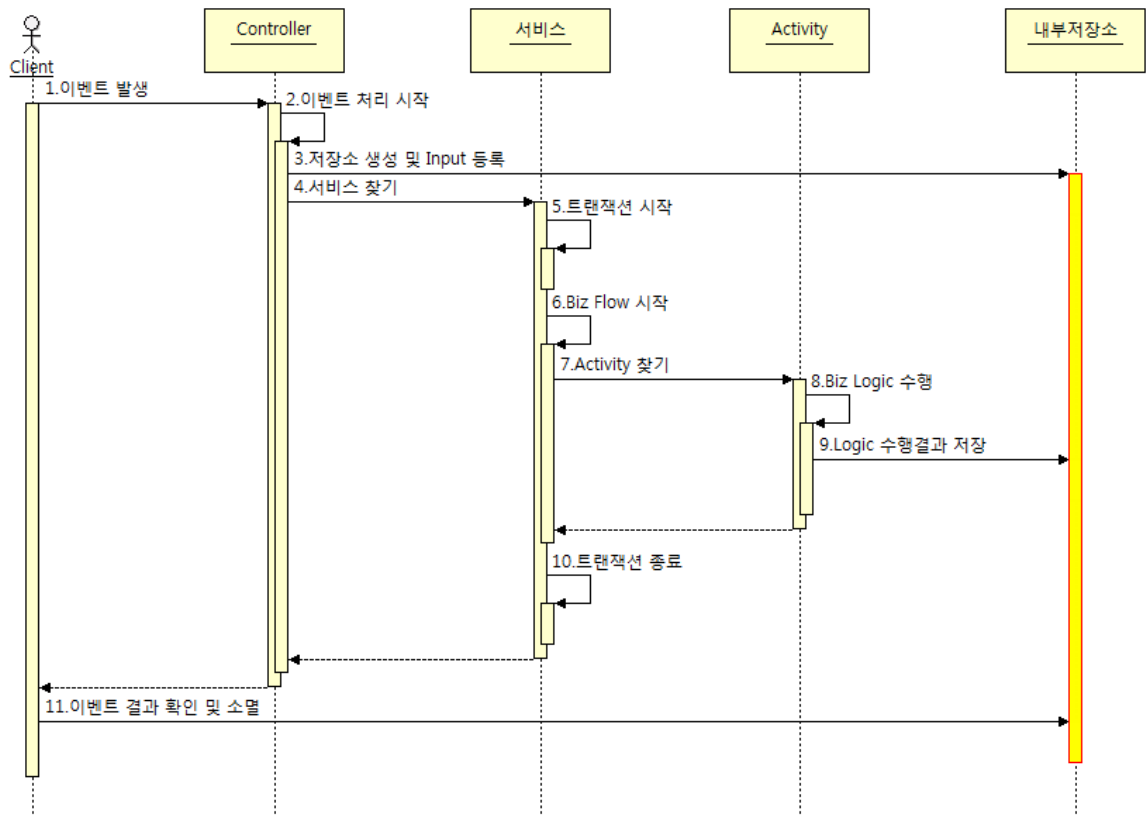
```

GlueContext ctx ...;
Object value = ctx.get(key);
  
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
String[] param = (String[])ctx.get(key, WebContentType.HTTP_REQUEST_PARAM);
Object attr = ctx.get(key, WebContentType.HTTP_REQUEST_ATTR);
```

Glue Framework 에서 실행가능한 최소단위는 GlueService 이다. GlueService 는 Activity 로 구성되었으며, 각 Activity 들에서 사용하는 Data 는 내부저장소를 통해 제공된다. 다음은 이벤트가 발생한 이후의 개략적인 순서와 내부저장소와의 관계이다.



[그림 39] 내부저장소

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

5. Service

Glue Service 는 독립적인 Business 의 한 기능을 수행하는 실행 단위이다. Activity Diagram 으로 작성해서 Glue Plug-In 을 이용해 실행시 사용할 수 있는 xml 파일로 생성한다. Glue Activity Diagram 은 보통 재사용이 가능한 단위로 작성하며, Glue Service 의 변경을 Activity Diagram 에서 작업해서 xml 파일을 재생성 하도록 한다.

Glue Service 는 실행 단위이며, Service 별로 Transaction 이 관리된다.

Glue Service 는 아주 작은 하나의 기능단위로 할 수도 있고 반대로 모든 Business 를 하나의 Service 로도 처리할 수 있지만 이것은 설계자의 목적에 따라 정의 될 수 있다.

이러한 Activity Diagram 은 다음과 같이 {pgm}-service.xml 로 생성되어, 실행시 사용된다.

```
<?xml version="1.0" encoding="UTF-8"?>
<service name="sample-service" initial="Router"
xmlns="http://www.poscoict.com/glueframework/service">
  <transaction-manager id="test-tx" commit="true"/>
  <activity name="Router"
    class="com.poscoict.glueframework.biz.activity.GlueDefaultRouter">
    <transition name="update" value="수정"/>
    <transition name="delete" value="삭제"/>
    <transition name="insert" value="등록"/>
    <transition name="success" value="end"/>
    <transition name="find" value="조회"/>
    <transition name="custom" value="SeachEmp-SQL"/>
  </activity>
  <activity name="등록"
    class="com.poscoict.glueframework.biz.activity.GlueJdbcInsert">
    <transition name="success" value="조회"/>
    <property name="sql-key" value="sample.emp.insert"/>
    <property name="param-count" value="4"/>
    <property name="chk-name" value="chk_insert"/>
    <property name="param0" value="empno_insert"/>
    <property name="param1" value="ename_insert"/>
    <property name="dao" value="test-dao"/>
    <property name="param2" value="sal_insert"/>
    <property name="param3" value="deptno_insert"/>
  </activity>
  <activity name="삭제"
    class="com.poscoict.glueframework.biz.activity.GlueJdbcDelete">
    <transition name="success" value="조회"/>
    <property name="sql-key" value="sample.emp.delete"/>
    <property name="dao" value="test-dao"/>
    <property name="chk-name" value="chk"/>
    <property name="param-count" value="1"/>
    <property name="param0" value="EMPNO"/>
  </activity>
  <activity name="수정"
    class="com.poscoict.glueframework.biz.activity.GlueJdbcModify">
    <transition name="success" value="조회"/>
    <property name="param-count" value="3"/>
    <property name="chk-name" value="chk"/>
    <property name="param0" value="SAL"/>
    <property name="param1" value="ENAME"/>
  </activity>
</service>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

    <property name="param2" value="EMPNO"/>
    <property name="dao" value="test-dao"/>
    <property name="sql-key" value="sample.emp.update"/>
  </activity>
  <activity name="조회"
    class="com.poscoict.glueframework.biz.activity.GlueJdbcSearch">
    <transition name="success" value="end"/>
    <property name="result-key" value="EmpList"/>
    <property name="sql-key" value="sample.emp.select"/>
    <property name="dao" value="test-dao"/>
    <property name="param-count" value="1"/>
    <property name="param0" value="deptno"/>
  </activity>
  <activity name="SeachEmp-SQL" class="sample.activity.SearchEmp">
    <transition name="success" value="end"/>
  </activity>
</service>

```

Service 를 구현한다는 것은 Service 자체를 구현하는 것이 아니라 Service 에 속한 각각의 Activity Class 를 구현하는 것이다.

이번 장에서는 Activity 에 구현되어야 할 내용과 함께 Glue Framework 에서 제공하는 기능과 사용 가능한 기술을 설명한다.

5.1 transaction-manager

Glue 에서 제공하는 Transaction Manager 는 기본적으로 GlueTransactionManager 을 상속받아 구현하였으며 구현체로 제공하고 있는 GlueDataSourceTransactionManager 의 경우에는 Spring 에서 제공하는 DataSourceTransactionManager 를 상속받아 만들어졌으므로 Spring 에서 제공하는 TransactionManager 관련 기능들과도 연동이 가능하다.

하나의 Service 에 등록된 transaction-manager 는 해당 Service 에서 처리할 Transaction List 이다. 하나의 Transaction 은 같은 이름을 사용하는 DataSource 또는 JDBC URL 로 commit/rollback 을 처리할 수 있는 단위이다. 즉 같은 DB 를 의미한다.

<service 일부>

```
<transaction-manager id="tx1" commit="true"/>
```

항목 설명 :

- id : Spring 의 applicationContext.xml 에 정의 되어 있는 Transaction Manager 의 ID
- commit: 해당 Transaction 을 commit 할 지에 대한 여부. SubService 의 경우 MainService 의 Transaction 을 사용하고 실제 Commit 을 하지 않을 경우 "false"로 정의 하거나 service 에 Transaction 을 등록하지 않아야 한다.

<applicationContext.xml 일부>

```

<bean id="tx1"
  class="com.poscoict.glueframework.transaction.GlueDataSourceTransactionManager">
  <property name="dataSource">

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

        <ref local="dataSource"/>
    </property>
</bean>

```

GlueBizController 는 service 를 실행할 때 service 에 등록되어 있는 Transaction 들을 Start 하며 정상 종료 하였을 경우에는 start 된 Transaction 들을 Commit 하고 Exception 이 발생한 경우에는 start 된 Transaction 들을 Rollback 한다.

Transaction 의 관리(start,commit,rollback)은 Glue 에 맡기는 것(service 시작시 start , 종료시 commit 혹은 rollback)을 권장하며 Service 실행 중에 Commit 혹은 Rollback 동작이 꼭 필요한 경우에는 아래와 같이 GlueActivity 에서 제공하는 commitTransaction 혹은 rollbackTransaction 메서드를 사용하도록 한다.(transactionMgrKey:commit 혹은 rollback 할 transaction manager 의 bean id)

```

public class GlueCustom extends GlueActivity<GlueContext> {
    @Override
    public String runActivity( GlueContext ctx )
    {
        . . .
        transactionMgrKey = . . .
        try
        {
            this.commitTransaction(transactionMgrKey);
        } catch ( Exception ex ) {
            this.rollbackTransaction(transactionMgrKey)
        }
        . . .
    }
}

```

5.2 Activity 구현

Activity 는 Reuse Activity 와 Custom Activity 로 구분된다. Reuse Activity 는 activity 의 property 를 설정하는 것으로 코딩을 요구 하지 않는다. Custom Activity 는 반드시 com.poscoict.glueframework.biz.activity.GlueActivity Class 를 상속받은 Class 여야 하며, GlueActivity 는 abstract Class 이기 때문에 "public String runActivity(GlueContext ctx)" Method 를 반드시 구현하여야 한다.

※ 주의사항

생성자 overriding : Activity Class 의 생성 시기는 최초 Service Loading 시에 한번만 생성되기 때문에 별도로 생성자를 구현하지 않는 것이 좋다. 즉 생성자는 Request 마다 생성되는 것이 아니고 Service Loading 시 한번 생성되고 재사용 된다. 따라서 생성자를 생성하지 않는 것이 좋다.

Member 변수 선언 : Activity 의 Member 변수는 선언하지 않도록 한다. Activity 는 한 Request 에서만 존재 하는 것이 아니라 Service 단위로 존재하기 때문에 같은 Service 를 사용하는 모든 사용자 및 App 에서 동일한 Instance 를 사용하게 되기 때문에 해당 Activity 의 Member 변수도 마찬가지로 공유하며 Member 변수를 잘못 사용할 경우 Memory Leak 을 발생시킬 수 있기 때문에 Custom Activity 작성 시 Member 변수는 사용하지 않을 것을 권고한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

<Sample - Activity>

```
//GlueActivity 를 extends 한다.
public class SampleActivity extends GlueActivity
{
    //Member 변수를 선언하지 않는다.
    //runActivity 에 business logic 를 구현한다.
    public String runActivity(GlueContext ctx)
    {
        // biz logic 구현
        // dao 는 applicationContext.xml 에 정의되어 있다.
        GlueGenericDao dao = this.getDao("testdao");
        GlueParameter<List> whereParam = new GlueParameter<List>();
        String paramCnt = this.getProperty ("param-count");
        if (paramCnt != null)
        {
            List<String> list = new ArrayList<String>();
            int count = Integer.parseInt(paramCnt);
            for (int i = 0; i < count; i++)
            {
                String param = this.getProperty("param" + i);
                //jsp page Request Parameter 는
                //모든 Input Data Type 이 String [] 형태임.
                String [] value = (String[])ctx.get(param);
                list.add(value[0]);
            }
            whereParam.setParameter(list);
            List rowList = dao.find("emp.select",whereParam);
            ctx.put("EmpListResult", rowList);
            // service 에 지정된 transition name
            return GlueBizControlConstants.SUCCESS;
        }
    }
}
```

5.2.1 Activity Parameter 이용

Activity Parameter(Activity Property)는 service.xml 에서 해당 Activity 의 Parameter 를 설정 하고 그 값을 이용 한다.

Reuse Activity 는 property 가 미리 정의되어 있으며 다음 chapter 를 참고한다.

Custom Activity 는 property 를 service 에 정의 하고 Activity Class 에서 이를 사용할 수 있도록 구현한다. Runtime 시에 property 를 추가하거나 제거할 수 있으나. 해당 Service 를 사용하는 모든 App 와 User 에게 적용되기 때문에 신중하게 사용하여야 한다.

다음은 service.xml 내용의 일부이다.

<Sample - service.xml>

```
<activity name="SearchResult"
    class="sample.activity.SampleActivity">
    <transition name="success" value="end" />
    <property name="dao" value="testdao" />
</activity>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
<property name="sql-key" value="emp.select" />
<property name="param0" value="DeptnoP" />
<property name="param1" value="EmpnoP" />
<property name="param-count" value="2" />
</activity>
```

다음은 위에 선언된 property 의 name 과 value 값을 가져 오는 Java Code 의 일부이다.

<Sample - Java Code>

```
String paramCnt = this.getProperty ("param-count");
String queryKey = this.getProperty ("sql-key");
String deptno[] = (String[])ctx.get(this.getProperty("DeptnoP"));
```

5.2.2 DAO Class 의 Handling

DAO(Data Access Obejct)는 얻는 방법을 제공하며, applicationContext.xml 에 해당 설정 정보가 정의되어 있다.

Activity Class 에서 DAO Class 는 다음과 같이 접근 할 수 있다.

<Sample - Java Code>

```
GlueGenericDao dao = this.getDao("testdao");
GlueGenericDao dao = this.getDao(this.getProperty("dao"));
```

여기서 getDao()의 Parameter 는 applicationContext.xml 의 Bean ID 이다.

<Sample - applicationContext.xml>

```
<bean id="testdao" class=" com.poscoict.glueframework.dao.jdbc.GlueJdbcDao">
  <property name="dataSource"><ref local="dataSource"/></property>
  <property name="queryManager"><ref local="queryManager"/></property>
</bean>
```

applicationContext.xml 의 dao 가 참조하는 dataSource 는 JNDI 이름으로 연결하는 방식과 JDBC URL 을 이용하는 2 가지 방식이 있다. Tomcat, OC4J 와 같은 Container 없이 Local(standalone)에서 Test 하기 위해서는 JDBC URL 방식을 이용하고, Container(WAS)위에서 실행한다면 JNDI 방식을 이용한다.

<Sample - applicationContext.xml>

```
<!-- JDBC URL -->
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
  destroy-method="close">
  <property name="driverClassName"
    value="oracle.jdbc.driver.OracleDriver"/>
  <property name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
  <property name="username" value="edu30"/>
  <property name="password" value="edu30"/>
</bean>
<!-- JNDI -->
<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName" value="jdbc/rw/TestDS"/>
  <property name="resourceRef" value="false"/>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

</bean>

5.3 Reuse Activity

Reuse Activity 는 Service 부분에서 반복적으로 재사용 가능한 공통적인 기능을 만들어 쉽게 활용하기 위해 제공한다.

Reuse Activity 는 향후 추가 삭제 될 수 있으며 각 Site 에서 별도의 Reuse Activity 를 만들어 사용 할 수도 있다.

더 자세한 내용은 Glue F/W 의 Java-Doc 문서를 참고하면 최신의 Reuse Activity 목록 및 사용법을 얻을 수 있다.

5.3.1 Jdbc Search Activity

조회용 SQL 의 Where Clause 와 화면의 Request Parameter 를 자동으로 Binding 하여 그 결과를 GlueContext 에 저장한다.

GlueJdbcSearch Class 는 특정 Select Query 를 실행하고 그 결과값을 저장 하게 된다.

Web 화면과 NonUI 공통으로 사용되고 Binding Parameter Type 은 Web 인 경우 String []의 {0} 번째로 Binding 하고 Web 이 아닌 경우는 해당 Object 를 Binding 한다.

Class 와 그 Property 는 다음과 같다.

Class : com.poscoict.glueframework.biz.activity.GlueJdbcSearch

property	설명	필수
dao	applicationContext.xml 의 DAO id	Y
sql-key	xxx-query.glue_sql 의 query id	Y
param-count	Binding 할 개수 (select * from emp where deptno=?)의 "?" 수	
param#	param0,param1... 으로 param-count 개수만큼 추가함. Binding Value ("?"와 순서 일치 하여야 함)	
param-bindings	binding 에 사용되는 값과 Mapping 되는 Key bindName=ctxName[bindName=ctxName] 로 정의. bindName 의 named query(select * from EMP where DEPTNO=:deptno) 의 named param 이름이며, ctxName 은 Binding Value 에 해당하는 ctx key. ctx 의 key 가 binding variable name 과 같은 경우 생략 가능함. 대소문자 분함.	
result-key	Context 에 담기는 Query 수행 결과 Key. Default : <sql-key>_resultList	
cache-key	기존 Query 수행결과 존재유무 확인후 Query 를 수행하도록 함	
cache-region	기존 Query 수행결과 존재유무 확인후 Query 를 수행하도록 함. Default : _default-region	

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

cache-manager	기존 Query 수행결과 존재유무 확인후 Query 를 수행하도록 함. Default : cacheManager	
---------------	---	--

[표 2] GlueJdbcSearch

Activity 의 XML 은 다음과 같이 정의 된다.

```
<activity name="Find"
    class="com.poscoict.glueframework.biz.activity.GlueJdbcSearch">
    <property name="dao" value="testdao" />
    <property name="sql-key" value="emp.select" />
    <property name="param-count" value="1" />
    <property name="param0" value="DeptnoP" />
    <property name="result-key" value="EmpList" />
    <transition name="success" value="end" />
</activity>
```

5.3.2 Jdbc Modify Activity

Update Query 의 set 절과 Where 절의 Parameter 를 화면의 Request parameter 와 자동으로 Bind 하여 Update 를 구현 하는 기능을 한다.

GlueJdbcModify Class 는 "update emp set sal=?, ename=? where empno=?" 와 같이 Update SQL 을 수행하는 Activity 이다.

Web 화면과 NonUI 공통으로 사용되고 Binding Parameter Type 은 Web 인 경우 String []의 {0}번째로 Binding 하고 Web 이 아닌 경우는 해당 Object 를 Binding 한다.

여기서 chkname 이 있는 경우 Web 의 CheckBox 에 Check 된 수만큼 반복 하고 없는 경우는 Context 에서 바로 String[] 로 get 하여 Binding 한다

NonUI 의 경우 Data Type 이 String[]가 아닌 경우 바로 해당 Object 로 Binding 한다.

결과값은 Update 한 Count 를 결과로 저장할 수 있다.

Class 와 그 Property 는 다음과 같다.

Class : com.poscoict.glueframework.biz.activity.GlueJdbcModify

property	설명	필수
dao	applicationContext.xml 의 DAO id	Y
sql-key	xxx-query.glue_sql 의 query id	Y
param-count	Binding 할 개수 (update emp set ename=? job=? where empno=?)의 "?" 수	
param#	param0,param1... 으로 param-count 개수만큼 추가함. Binding Value ("?"와 순서 일치 하여야 함)	
param-bindings	binding 에 사용되는 값과 Mapping 되는 Key bindName=ctxName[bindName=ctxName] 로 정의. bindName 의 named query(select * from EMP where DEPTNO=:deptno) 의 named param 이름이며, ctxName 은 Binding Value 에 해당하는 ctx key. ctx 의 key 가 binding variable name 과 같은 경우 생략 가능함. 대소문자 분함.	

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

result-key	Context 에 담기는 Query 수행 결과(수정 record 수) Key. Default : <sql-key>_updateCnt	
is-audit	GlueParameter 를 생성할때 GlueContext 에 auditAttributes 가 설정되어 있다면 해당 값을 GlueParameter 에 설정(자세한 내용은 16. Audit 참고)	
chk-name	화면의 CheckBox ID(HttpRequest 의 parameter name). <property name="chk-name" value="chk" /> 와 같이 "chk" 가 선언 되어 있다면, <input type="checkbox" name="chk">의 value 값에 따라 Looping 처리를 한다. chk 의 value 는 0,1,2,...,n 으로 부여되며 checked 된 값만 Context 에 담긴다.	
list-key	SQL 문에 Binding 할 Data Context Key. <property name="list-key" value="dataList" /> 와 같이 "dataList" 가 선언 되어 있다면, 이전 Activity 에서 List(Map<name, value>) 구조의 Data 를 생성해서 Context 에 담는다. [sample java code] <pre> List dataList = new ArrayList(); Map data1 = new HashMap(); data1.put("EMPNO", "1111"); data1.put("ENAME", "SCOTT"); data1.put("DEPTNO", "10"); dataList.add(data1); Map data2 = new HashMap(); data2.put("EMPNO", "2222"); data2.put("ENAME", "JANE"); data2.put("DEPTNO", "10"); dataList.add(data2); ctx.put("dataList", dataList); </pre>	

[표 3] GlueJdbcModify

Activity 의 XML 은 다음과 같이 정의 된다.

```

<activity name="Modify"
    class="com.poscoict.glueframework.biz.activity.GlueJdbcModify">
    <property name="dao" value="testdao" />
    <property name="sql-key" value="emp.update" />
    <property name="param-count" value="3" />
    <property name="param0" value="SAL" />
    <property name="param1" value="ENAME" />
    <property name="param2" value="EMPNO" />
    <property name="result-key" value="updateCnt" />
    <transition name="success" value="Find" />
</activity>

```

5.3.2.1 Web 화면 예제

다음 Sample 은 화면에 chkname 에 해당하는 checkbox 의 checked 한 value 값을 loop 로 돌면서 update query 문을 수행한다.

<Sample - JSP >

```

<input type="checkbox" name="chk" value=0 checked>
<input type="checkbox" name="chk" value=1 checked>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
<input type="checkbox" name="chk" value=2>
<input type=hidden name=EMPNO value="1111">
<input type=hidden name=EMPNO value="2222">
<input type=hidden name=EMPNO value="3333">
<input type=text name=ENAME value="홍길동">
<input type=text name=ENAME value="스미스">
<input type=text name=ENAME value="스미레">
```

<Sample - Query>

```
<query id="emp.update">
  <![CDATA[
    update EMP set ENAME=? where EMPNO=?
  ]]>
</query>
```

<Sample - Service>

```
<activity name="Modify"
  class="com.poscoict.glueframework.biz.activity.GlueJdbcModify">
  <property name="sqlkey" value="emp.update" />
  <property name="dao" value="testdao" />
  <property name="param-count" value="2" />
  <property name="param0" value="ENAME" />
  <property name="param1" value="EMPNO" />
  <property name="resultkey" value="UpdateCount" />
  <property name="chkname" value="chk" />
  <transition name="success" value="Find" />
</activity>
```

5.3.2.2 NonUI 예제

NonUI 의 경우 화면과 같은 chkname 이 없기 때문에 list-key property 를 이용한다. list-key property 를 사용하기 위해서는 이전 Activity 에 필요한 정보를 설정하여야 한다. list-key property 의 value 를 key 로 GlueContext 에 담길 내용은 update 문 bind 에 사용된 param 정보들이다. Sample Code 와 같이 HashMap 이 필요한 만큼 ArrayList 에 담는다.

만일 list-key property 가 없다면 Context 에서 "SAL","ENAME"등을 Key 로 Value 를 Binding 하게 되고 한건만 Update 하게 된다.

<Sample - Query>

```
<query id="emp.update">
  <![CDATA[
    update EMP set ENAME=? where EMPNO=?
  ]]>
</query>
```

<Sample - Service>

```
<activity name="SetUpdateList" class="sample.activity.SetEmpUpdateList">
  <transition name="success" value="Modify" />
</activity>
<activity name="Modify"
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

        class="com.poscoict.glueframework.biz.activity.GlueJdbcModify">
        <property name="sqlkey" value="emp.update" />
        <property name="dao" value="testdao" />
        <property name="param-count" value="2" />
        <property name="param0" value="ENAME" />
        <property name="param1" value="EMPNO" />
        <property name="resultkey" value="UpdateCount" />
        <property name="list-key" value="updateList" />
        <transition name="success" value="Find" />
    </activity>

```

<Sample – Custom Activity 인 SetEmpUpdateList.java>

```

List updateData = new ArrayList();
Map data1 = new HashMap();
data1.put("ENAME", "Young");
data1.put("EMPNO", "1111");
Map data2 = new HashMap();
data2.put("ENAME", "Koung");
data2.put("EMPNO", "2222");
updateData.add(data1);
updateData.add(data2);
ctx.put("updateList", updateData);

```

5.3.3 GlueParseMessage

GlueParseMessage Activity 는 TC(String)을 Layout 으로 Parse 하여 GlueMessage Instance 를 생성하는 기능을 수행한다.

전제 사항:

Context 에 "ReceiveTC" 를 key 로 TC 가 저장 되어 있거나 Context 에 GlueMessage 가 있어야 한다.

즉 ctx.getMessage()가 null 아니거나 ctx.get("ReceiveTC");가 null 이 아니어야 한다.

Class 와 그 Property 는 다음과 같다.

Class : com.poscoict.glueframework.biz.activity.GlueMessageParse

property	설명	필수
layout	applicationContext.xml 의 layoutManager bean id.	Y
id-length	"ReceiveTC"에 해당하는 String 의 substring 기준. [default] : 8	
interface-type	tc/file [default] : tc	
charset	ParseType 이 byte 일 경우 사용됨. 고정값.	
charset-name	ParseType 이 byte 일 경우 사용됨. 동적값.	

[표 4] GlueMessageParse

Activity 의 XML 은 다음과 같이 정의 된다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
<activity name="Parse"
    class="com.poscoict.glueframework.biz.activity.GlueMessageParse">
    <transition name="success" value="SetMessageKey" />
    <property name="layout" value="layoutInXml" />
</activity>
```

applicationContext.xml 의 LayoutManager 는 다음과 같이 정의 가능하다.

```
<!-- XML Layout Manager -->
<bean id="layoutInXml"
    class="com.poscoict.glueframework.message.layout.GlueXmlMessageLayout"/>
<!-- DB Layout Manager -->
<bean id="layoutInDB"
    class="com.poscoict.glueframework.message.layout.GlueDBMessageLayout">
    <property name="dao" ref="masterdao"/>
    <property name="queryId" value="get_msg_layout_query"/>
</bean>
```

Parse Type 지정 방법: 별도로 지정하지 않으면 Byte Type 으로 parse 한다. Parsing Type 은 glue.properties 를 통해 지정한다. msg.parsing.type 이라는 property 를 사용하면 그 값은 byte 또는 string 이 가능하다.

```
msg.parsing.type=byte
```

byte parsing 시 String 을 byte[] 로 Encoding 우선순위는 다음과 같다.

charset-name property > charset property > jvm default

5.3.4 GlueCreateMessage

GlueCreateMessage Activity 는 GlueMessage 객체를 정의된 Layout 에 근거해 String 객체로 생성한다. 생성된 String 객체는 List<String> 구조로 Context 에 담긴다. GlueContext 에는 TC 를 생성할 GlueMessage 객체가 있어야 한다.

Class 와 그 Property 는 다음과 같다

Class : com.poscoict.glueframework.biz.activity.GlueMessageCreate

property	설명	필수
layout	applicationContext.xml 의 layoutManager bean id.	Y
result-key	Context 에 Map 형태로 결과를 담을 Key. Data 구조는 Map<String, List<String>>. 즉, Map 에는 ArrayList 가 담겨 있고 그 Key 값은 TCID 이다. 하나의 결과 또는 같은 종류의 여러개의 결과가 ArrayList 로 저장됨.	Y
message-key	TC 를 생성할 GlueMessage 객체 messag-key 에 담길 수 있는 Data Type 은 다음과 같다. GlueMessage GlueMessage [] List<GlueMessage> : GlueMessage 가 담겨 있어야 한다	Y

[표 5] GlueCreateMessage

Activity 의 XML 은 다음과 같이 정의 된다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<activity name="CreateMessage"
    class="com.poscoict.glueframework.biz.activity.GlueMessageCreate">
    <transition name="success" value="PrintResult" />
    <property name="result-key" value="tcList1" />
    <property name="layout" value="layout" />
    <property name="message-key" value="message1" />
</activity>

```

5.3.5 Routing Activity

Router 의 경우 두 종류의 Class 를 지정할 수 있다. 하나는 GlueDefaultRouter 이고 다른 하나는 GlueDefaultRouter 를 상속받은 Custom Router Class 이다.

```

<activity name="EventRouter"
    class="com.poscoict.glueframework.biz.activity.GlueDefaultRouter">
    <transition name="find" value="FindData"/>
    <transition name="update" value="UpdateData"/>
    <transition name="delete" value="DeleteData"/>
</activity>

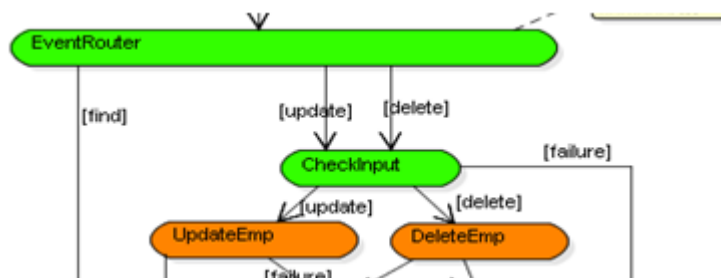
```

위의 경우 F/W 에서 제공하는 Default Router(GlueDefaultRouter)를 지정 하였다.

Router 는 Context(GlueContext)에 Transition Name 이 존재 하면 해당 Activity 로 Routing 하는 역할을 한다.

즉 위 예제에서 Context 에 find=x 로 저장되어 있다면 Router 는 FindEmp Activity 를 실행 하게 된다.

주의 !-> Context 에 하나 이상의 Event 에 동시에 값이 존재한다면(예 : find=x, update=y) Error 가 발생



[그림 40] DefaultRouter

Router 의 Transition 에는 해당 Router Activity 가 처리할 Event 를 모두 정의 하여야 한다. 또한 위 그림처럼 EventRouter 에서 사용되었던 Event(update,delete)는 다음 Router 에서 사용 될 수 있다.

위 예제처럼 Default Transition (“success”)가 존재 하지 않는 다면 위 Event(find,update,delete)가 발생하지 않는 경우 Error 처리 된다.

Success Transition 이 존재 한다면 Event 발생하지 않는 경우 “success” 로 분기 하게 된다

5.3.6 Sub Service 이용 (Service 재사용)

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Sub Service 를 이용하기 위해서는 Activity 에 GlueSubBizControlActivity 를 등록하여 사용하여야 한다. Sub Service 는 Main Service 에서 하나의 Activity 로 동작하게 된다. 사용 방법은 다음과 같이 Service.xml 에 Activity 를 정의하여 사용한다.

```
<activity name="CallSubController"
    class="com.poscoict.glueframework.biz.activity.GlueSubService">
    <transition name="success" value="HandleError"/>
    <property name="service-name" value="SubService-service"/>
    <property name="new-transaction" value="true"/>
</activity>
```

위에서 Activity Class 는 반드시 “com.poscoict.glueframework.biz.activity.GlueSubService” 를 지정하여야 한다.

property	설명	필수
service-name	Service Name.	Y
new-transaction	true, false	

[표 6] GlueSubService

ServiceName 은 필수 항목으로 Sub Service 의 명칭을 Value 로 넣어야 한다.

new-transaction 항목은 Main 과 Sub Service 가 별개의 Transaction 을 가지고 있는 경우 true 로 하여야 하고 그렇지 않고 Main 의 Transaction 을 공유한다면 false 로 하여야 한다.

Sub Service 를 사용하기 위해서는 In/Out 에 대해 명확하게 인지 하고 있어야 한다. Main Service 와 Sub Service 의 Data 교환은 GlueContext 를 통해 이루어 지기 때문에 Input 으로 사용될 Key/Value 와 Output 으로 사용될 Key/Value 에 대해 상호간에 명확하여야 한다.

주의 사항:new transaction 이 true 인 경우에는 Main Service 와 별도의 transaction 으로 동작하기 때문에 Main 과 상관없이 Transaction 이 commit/rollback 된다.

case	new-transaction = true	new-transaction = false 또는 미정의
Sub Servicet 실행 후 Mian 에서 Exception 발생하는 경우	Main 에서 실행된 부분은 Rollback 되고 Sub Service 에서 실행된 부분은 이미 Commit 됨	Main 과 Sub service 에서 실행된 부분이 모두 Rollback 됨

[표 7] new-transaction 설정에 따른 commit, rollback 범위

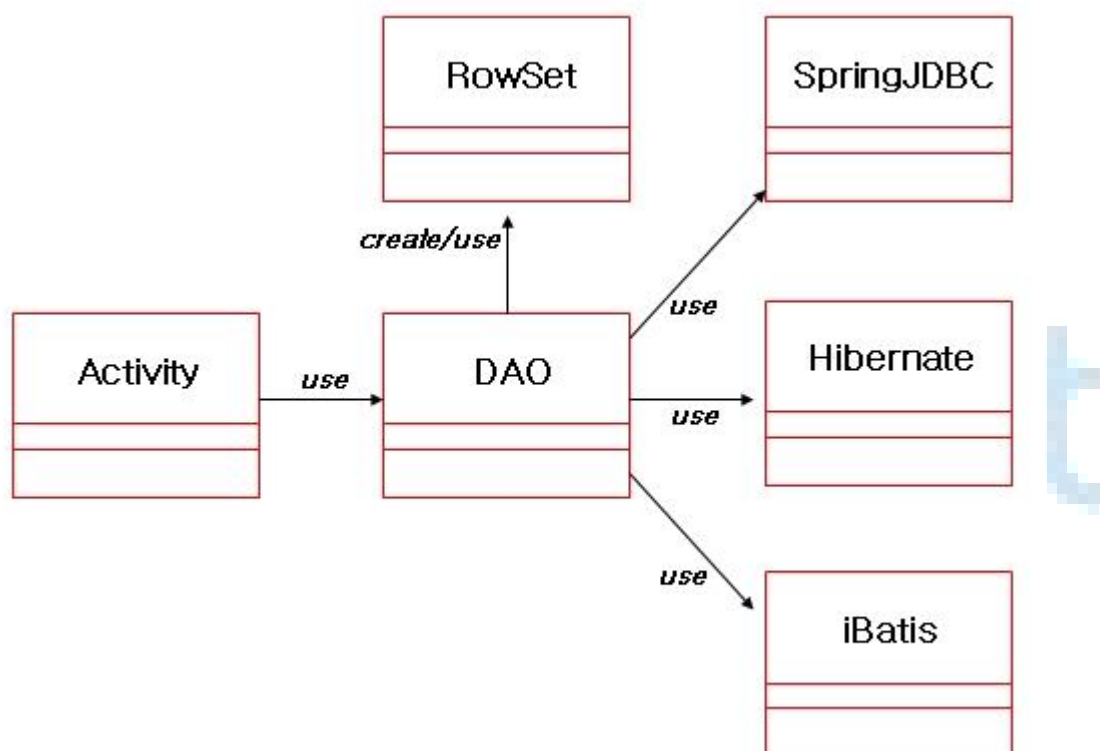
	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

6. Data Access Object(DAO)

DAO는 Data Access Object의 약자로 데이터 소스(일반적으로 Database, LDAP 또는 파일)에 대한 캡슐화 및 access를 제공합니다.

DAO는 J2EE를 기반으로 하는 일반적인 프로젝트에서 자주 사용되는 디자인 패턴입니다. DAO는 메소드를 호출하는 오브젝트에 대하여 추상화 된 데이터 소스로서의 기능을 제공하며, 데이터에 대한 투명성(transparency)을 제공합니다. 즉 구체적인 데이터 소스에 대한 동작을 DAO가 처리함으로써 DAO를 호출하는 오브젝트에게 데이터 소스에 대한 상세한 정보를 숨기게 하여 상호 의존도를 줄이는 역할을 합니다. 궁극적으로는 데이터 액세스 로직과 비즈니스 로직을 분리하자는 의도로 DAO 패턴을 사용합니다.

DAO는 그림과 같이 SpringJDBC, Hibernate, iBatis를 이용해서 Activity에서 사용됩니다.



[그림 41] Data Access Object

이런 구조에서 DAO가 없으면, 데이터 소스인 SpringJDBC, Hibernate, iBatis와 관련된 코드가 Activity 클래스에 사용되게 되며, 상세한 동작을 Activity가 알고 있어야 한다는 것을 의미합니다.

Glue Framework 4.0에서는 3종류의 DAO를 제공합니다. 이러한 DAO는 applicationContext.xml 파일에 빈(bean)으로 설정합니다. DAO종류별 설정 및 사용법은 다음 장에 설명 됩니다.

6.1 GlueGenericDao

GlueGenericDao는 데이터베이스 CRUD(create:insert, read:select, update, delete) 오퍼레이션을 추상화하여 제공하는 Data Access Object 인터페이스입니다. GlueGenericDao의 메소드는 다음과 같습니다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

return	method	설명
List	find(String name) find(String name, GlueParameter param)	name 를이용해서 Data 를 조회한다.
int	insert(String name) insert(String name, GlueParameter param)	name 을 이용해서 Data 를 추가/등록한다.
int	update(String name) update(String name, GlueParameter param)	name 을 이용해서 Data 를 수정/갱신한다.
int	delete(String name) delete(String name, GlueParameter param)	name 을 이용해서 Data 를 삭제한다.

[표 8] GlueGenericDao 메소드

<applicationContext.xml 일부>

```
<bean id="test-dao" class= . . . >
</bean>
```

<Java Code 일부>

```
GlueParameter param = . . . ;
GlueGenericDao dao = . . . ;
List rowSet = dao.find("sample.emp.select");
int cnt = dao.insert("sample.emp.insert", param);
```

6.2 GlueJdbcDao

GlueJdbcDao 는 SpringJDBC 를 이용해서 추상화된 CRUD 오퍼레이션을 제공하는 구현체입니다. GlueJdbcDao 는 GlueGenericJdbcDao 인터페이스를 구현하였으며, GlueGenericJdbcDao 는 GlueGenericDao 를 확장하였으며, 그 메소드는 다음과 같습니다.

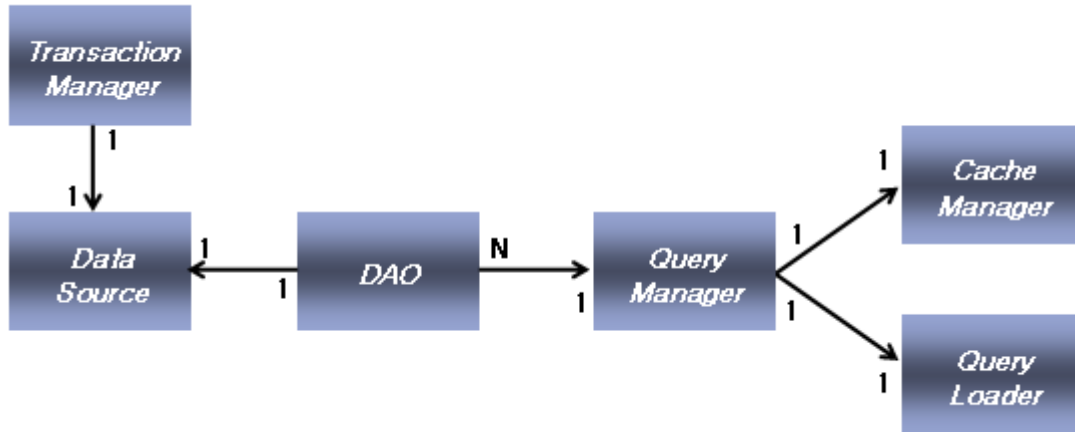
method	설명
find(String queryKey) find(String queryKey, GlueParameter param)	xxx-query.glue_sql 에 정의한 Query 로 DB 레코드를 조회/수정/삽입/삭제한다.
insert(String queryKey) insert(String queryKey, GlueParameter param)	
update(String queryKey) update(String queryKey, GlueParameter param)	
delete(String queryKey) delete(String queryKey, GlueParameter param)	
findByQueryStatement(String queryStmt) findByQueryStatement(String queryStmt, GlueParameter param)	프로그램 소스상에서 Query 를 직접 작성하여 DB 레코드를 조회/수정/삽입/삭제한다.
insertByQueryStatement(String queryStmt) insertByQueryStatement(String queryStmt, GlueParameter param)	
updateByQueryStatement(String queryStmt) updateByQueryStatement(String queryStmt, GlueParameter param)	
deleteByQueryStatement(String queryStmt) deleteByQueryStatement(String queryStmt, GlueParameter param)	

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

[표 9] GlueGenericJdbcDao 메소드

GlueGenericJdbcDao 의 메소드는 쿼리 파일을 사용하는 메소드와 직접 메소드 argument 에 쿼리 스트링을 기재 하는 두 가지 방법이 존재합니다. 구현체인 GlueJdbcDao 는 내부적으로 SpringJDBC 를 이용해 select/insert/update/delete 등의 SQL 을 실행합니다. 쿼리 관리 상의 편의성과 가독성 향상을 위해서 쿼리파일 사용을 권장하며, 자세한 사용법은 Glue JavaDoc 을 통해 제공합니다.

GlueJdbcDao 를 사용하기 위해서는 설정파일(applicationContext.xml)에 GlueJdbcDao 관련 빈(bean)을 설정해 주어야 합니다. GlueJdbcDao 는 DataSource, QueryManager 를 필요로 하며 그 관계는 다음 그림과 같습니다.



[그림 42] GlueJdbcDao 의 bean 들과의 관계

DataSource 는 DB 연결정보는 관리하는 빈(bean) 오브젝트이며, DAO 와 1:1 의 관계가 됩니다. 즉 사용하는 DataSource 가 2 개라면 DAO 도 2 개가 되어야 합니다. 이러한 빈(bean) 오브젝트는 Spring IoC 에 의해 싱글 인스턴스로 관리되며, 모두 thread-safe 한 오브젝트 입니다. 위 그림과 같이 DataSource 개수만큼 DAO 가 정의되며, 각 DAO 는 동일 QueryManager 를 사용합니다. 즉 QueryManager 는 1 개만 정의해서 각 DAO 에서 사용되도록 정의합니다. QueryManager 는 query id 에 해당하는 SQL 을 DAO 가 사용할 수 있도록 인터페이스를 제공하며, query id 에 해당하는 SQL 은 CacheManager 에 의해 캐싱되며, QueryLoader 를 통해 Query file 에 정의된 SQL 을 읽어들이니다. TransactionManager 또한 DAO 처럼 DataSource 와 1:1 관계입니다.

다음은 설정파일(applicationContext.xml)의 일부입니다.

```

<bean id="dataSource-1" class= . . ./>
<bean id="dataSource-2" class= . . ./>
<bean id="dao-1" class="com.poscoict.glueframework.dao.GlueJdbcDao">
    <property name="dataSource" ref="dataSource-1"/>
    <property name="queryManager" ref="queryManager"/>
</bean>
<bean id="dao-2" class="com.poscoict.glueframework.dao.GlueJdbcDao">
    <property name="dataSource" ref="dataSource-2"/>
    <property name="queryManager" ref="queryManager"/>
</bean>
<bean id="transactionManager-1"
    class="com.poscoict.glueframework.transaction.GlueDataSourceTransactionManager">
    <property name="dataSource" ref="dataSource-1"/>
</bean>
<bean id="transactionManager-2"
    class="com.poscoict.glueframework.transaction.GlueDataSourceTransactionManager">

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

    <property name="dataSource" ref="dataSource-2"/>
</bean>
<bean id="queryManager"
    class="com.poscoict.glueframework.dao.manager.GlueQueryManagerImpl">
    <property name="cacheManager" ref="cacheManager"/>
    <property name="queryLoader" ref="queryLoader"/>
</bean>
<bean id="cacheManager" class= . . ./>
<bean id="queryLoader" class= . . . />

```

빈(bean)들간의 관계성은 모두 자바 인터페이스를 기반으로 레퍼런싱 되고 있습니다. 그러므로, 실제 구현 된 소스 코드를 변경하지 않고도 빈 설정 파일만 수정 해주면 변경된 로직으로 작동하는 새로운 오브젝트를 각 빈들과 연관을 지을 수 있습니다. 쉽게 말해서 설정파일(applicationContext.xml) 재정의만을 통해서 새로 제작된 DAO, QueryManager 등을 손쉽게 교체할 수 있다는 것을 의미합니다. 결론적으로 유지보수성 및 확장성이 아주 뛰어납니다.

빈(bean)으로서의 GlueJdbcDao 가 갖는 속성(property)은 다음과 같습니다.

property	설명	필수
dataSource	참조할 DataSource 를 지정한다.	Y
queryManager	참조할 QueryManager 를 지정한다.	Y
columnManager	참조할 ColumnManager 를 지정한다.	N
typeCast	조회시 결과값을 GlueRowSet<GlueRow> 변환할지 여부를 설정한다. - true : List<Map>을 GlueRowSet<GlueRow>로 변환함. 기본 설정은 false 이다.	N
batchUpdateLimit		N

[표 10] GlueJdbcDao 의 property

6.2.1 Glue Query File

GlueJdbcDao 는 2 종류의 Query 를 사용할 수 있습니다.

구분	설명	SQL 예
기본 SQL	SQL 작성시 input 이 물음표(?)로 표현되며, SQL 실행시 input parameter 는 순서대로 입력되어야 한다.	SELECT * FROM EMP WHERE EMPNO=? AND SAL>?
Named SQL	SQL 작성시 input 에 콜론(:) 다음에 이름을 주어 SQL 실행시 동일 이름의 값으로 input parameter 을 입력한다.	SELECT * FROM EMP WHERE EMPNO=:name AND SAL>:sal

[표 11] GlueJdbcDao 에서 사용 가능한 SQL 유형

다음은 Query 파일(xxx-query.glue_sql)의 일부입니다. isNamed 속성을 통해 Query 유형을 구분할 수 있습니다.

```

<query id="sample01.emp.select" desc="EMP 조회" resultType="" isNamed="false">

```

```

    <![CDATA[
select EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO
from EMP
where DEPTNO=?
    ]]>
</query>
<query id="sample01.emp.update" desc="EMP 수정" resultType="" isNamed="false">
    <![CDATA[
update emp
set sal=?, ename=?
where empno=?
    ]]>
</query>
<query id="sample01.emp.delete" desc="EMP 삭제" resultType="" isNamed="false">
    <![CDATA[
delete from emp
where empno=?
    ]]>
</query>
<query id="sample01.emp.insert" desc="EMP 등록" resultType="" isNamed="false">
    <![CDATA[
insert into emp(empno, ename, sal, deptno)
values(?, ?, ?, ?)
    ]]>
</query>
<query id="sample02.emp.select" desc="EMP 조회" resultType="" isNamed="true">
    <![CDATA[
select EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO
from EMP
where DEPTNO=:deptno
    ]]>
</query>
<query id="sample02.emp.update" desc="EMP 수정" resultType="" isNamed="true">
    <![CDATA[
update emp
set sal=:sal, ename=:ename
where empno=:empno
    ]]>
</query>
<query id="sample02.emp.delete" desc="EMP 삭제" resultType="" isNamed="true">
    <![CDATA[
delete from emp
where empno=:empno
    ]]>
</query>
<query id="sample02.emp.insert" desc="EMP 등록" resultType="" isNamed="true">
    <![CDATA[
insert into emp(empno, ename, sal, deptno)
values(:empno, :ename, :sal, :deptno)
    ]]>
</query>
</queryMap>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

6.2.2 GlueParameter

GlueJdbcDao 의 실행하고자 하는 SQL 이 input parameter 값을 필요한다면, GlueParameter 를 통해 그 값을 전달합니다. 실행하는 SQL 에 따라 GlueParameter 를 구성합니다. 기본 SQL 을 사용하는 경우는 List 로 GlueParameter 를 구성하고, Named SQL 을 사용하는 경우는 Map 으로 구성합니다. 구성된 List/Map 객체는 setParameter() 를 이용하거나, 생성자를 이용합니다.

다음은 기본 SQL 일 경우 GlueParameter 를 구성하는 예입니다.

```
GlueParameter<List> param = new GlueParameter<List>();
List paramList = new ArrayList();
paramList.add( value1 );
paramList.add( value2 );
paramList.add( value3 );
paramList.add( value4 );
param.setParameter(paramList);
// GlueParameter<List> param = new GlueParameter<List>(paramList);
```

다음은 Named SQL 일 경우 GlueParameter 를 구성하는 예입니다.

```
GlueParameter<Map> param = new GlueParameter<Map>();
Map paramMap = new HashMap();
paramMap.put("paramName1", value2 );
paramMap.put("paramName2", value3 );
paramMap.put("paramName3", value4 );
paramMap.put("paramName4", value1 );
param.setParameter(paramMap);
// GlueParameter<Map> param = new GlueParameter<Map>(paramMap);
```

6.2.3 GlueColumnManager

ColumnManager 는 JdbcDao 를 통해 조회한 Data 의 ColumnDefinition 을 생성해서 관리합니다.

Dao bean 정의시 columnManager property 를 통해 다음과 같이 등록한다.

```
<bean id="dao" class="com.poscoict.glueframework.dao.GlueJdbcDao">
  <property name="dataSource" ref="dataSource"/>
  <property name="queryManager" ref="queryManager"/>
  <property name="columnManager" ref="columnManager"/>
</bean>
<bean id="columnManager"
  class="com.poscoict.glueframework.dao.manager.GlueColumnManager"/>
<bean id="dataSource" class= . . . />
<bean id="queryManager" class= . . . />
```

ColumnManager 의 기능을 사용하기 위해서는 Dao 의 find() 를 통해 실행되는 SQL 이 물음표(?)를 사용해야 한다.

ColumnManager 를 갖는 DAO 는 find()의 실행 결과를 List<Map> 구조에서 GlueRowSet<GlueRow>의 구조로 변환한다. GlueRowSet 에서 제공하는 getColumnDefs() 를 통해 select 문의 실행결과 각 column 의 Name, Type, Length, Precision, Scale 정보를 얻어올 수 있다.

```
GlueGenericDao dao = getDao("testdao");
GlueRowSet rowSet = dao.findByQueryStatement("select * from EMP");
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

GlueColumnDef[] def = rowSet.getColumnDefs();
while(rowSet.hasNext())
{
    GlueRow row = rowSet.next();
    for(int i=0, iz=def.length; i < iz; i++)
    {
        System.out.println("ColumnDef - Name : "+def[i].getName());
        System.out.println("ColumnDef - Type : "+def[i].getType());
        System.out.println("ColumnDef - Length : "+def[i].getLength());
        System.out.println("ColumnDef - Precision : "+def[i].getPrecision());
        System.out.println("ColumnDef - Scale : "+def[i].getScale());
        System.out.println("Data : "+row.getAttribute(def[i].getName()));
    }
}

```

6.2.4 Reuse & Custom Activity

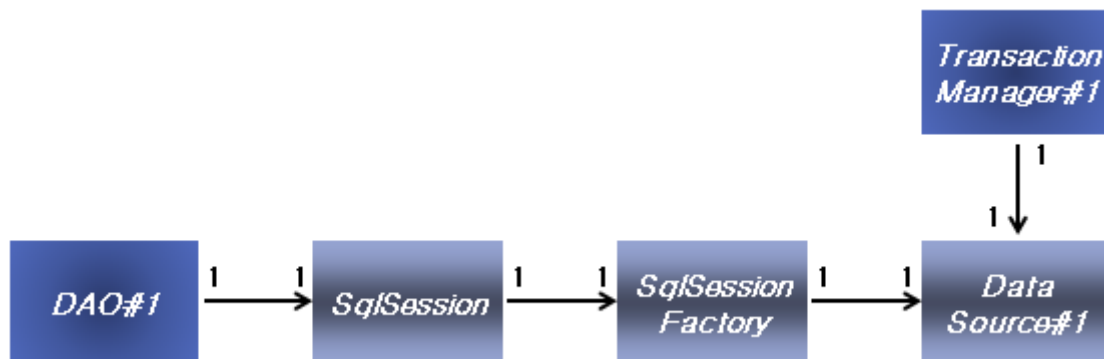
GlueJdbcDao 를 사용한 reuse Activity 에는 GlueJdbcSearch, GlueJdbcDelete, GlueJdbcInsert, GlueJdbcModify Activity 가 있습니다.

6.3 GlueMyBatisDao

GlueMyBatisDao 는 Mybatis 를 이용해서 추상화된 CRUD 오퍼레이션을 사용할 수 있는 Database Access Object 구현체입니다. GlueGenericDao 를 상속받아 구현된 Dao 클래스로 데이터베이스 CRUD(Create, Read, Update, Delete) 작업을 수행하는 Method 를 제공합니다..

본 가이드에서는 Glue 와 MyBatis 연계 관련된 기본적인 내용에 대해서만 다룰 것이므로, MyBatis 에 대한 자세한 내용은 MyBatis 공식 사이트의 매뉴얼을 참고하시기 바랍니다.

- 영문 사이트: <http://mybatis.github.io/mybatis-3/sqlmap-xml.html>
- 한글 사이트: <http://mybatis.github.io/mybatis-3/ko/sqlmap-xml.html>



[그림 43] GlueMyBatisDao 의 bean 들과의 관계

GlueMybatisDao 를 사용하기 위해서는 설정파일(applicationContext.xml)에 SqlSessionFactoryBean, SqlSessionTemplate, GlueMybatisDao 가 등록되어야 합니다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

- `SqlSessionFactoryBean` : MyBatis 에서는 `SqlSessionFactoryBuilder` 클래스를 사용하여 `session factory` 를 생성하는데 MyBatis-Spring 은 `SqlSessionFactoryBuilder` 클래스 대신 `SqlSessionFactoryBean` 클래스를 사용하여 생성합니다. `SqlSessionFactoryBean` 은 `dataSource`, `mapperLocation` property 를 통해, JDBC Datasource 정보와 MyBatis [mapper 파일](#)(쿼리 파일)의 위치를 지정합니다.
- `SqlSessionTemplate` : `SqlSessionTemplate` 클래스는 MyBatis-Spring 의 핵심 기능이며 MyBatis `SqlSessions` 관리 및 MyBatis SQL method 호출을 담당합니다. `SqlSessionTemplate` 클래스는 `SqlSessionFactory` 를 생성자의 인자로 받을수 있습니다.

다음은 설정파일(applicationContext.xml)의 일부입니다.

```
<bean id="dataSource-1" class="..." />
<bean id="dataSource-2" class="..." />
<bean id="sqlSessionFactory-1" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource-1" />
    <property name="mapperLocations" value="classpath*:mybatis/mapper/*.xml" />
</bean>
<bean id="sqlSessionFactory-2" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource-2" />
    <property name="mapperLocations" value="classpath*:mybatis/mapper/*.xml" />
</bean>
<bean id="sqlSession-1" class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg index="0" ref="sqlSessionFactory-1" />
</bean>
<bean id="sqlSession-2" class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg index="0" ref="sqlSessionFactory-2" />
</bean>
<bean id="dao-1" class="com.poscoict.glueframework.dao.mybatis.GlueMybatisDao">
    <property name="sqlSession" ref="sqlSession-1" />
</bean>
<bean id="dao-2" class="com.poscoict.glueframework.dao.mybatis.GlueMybatisDao">
    <property name="sqlSession" ref="sqlSession-2" />
</bean>
<bean id="transactionManager-1"
    class="com.poscoict.glueframework.transaction.GlueDataSourceTransactionManager">
    <property name="dataSource" ref="dataSource-1"/>
</bean>
<bean id="transactionManager-2"
    class="com.poscoict.glueframework.transaction.GlueDataSourceTransactionManager">
    <property name="dataSource" ref="dataSource-2"/>
</bean>
```

6.3.1 Mapper XML 파일

Mybatis Mapper XML(쿼리 파일) 은 `SqlSessionFactoryBean` 의 `mapperLocations` 에 정의된 Path 에 Mybatis 쿼리를 정의 합니다.

Mapper XML(SQL Map XML) 파일은 First Class(첫번째 요소)만을 갖습니다.

다음은 Mapper XML 예제입니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="emp.mapper">
  <select id="getEmp" parameterType="String"
    resultType="sample.vo.EmpVO">
    select * from EMP where DEPTNO=#{deptno}
  </select>
  <select id="getEmp2" parameterType="sample.vo.DeptVO"
    resultType="sample.vo.EmpVO">
    select * from EMP where DEPTNO=#{deptno}
  </select>
  <select id="getEmp3" parameterType="sample.vo.EmpVO"
    resultType="sample.vo.EmpVO">
    select * from EMP where DEPTNO=#{deptno} and sal>#{sal}
  </select>
  <select id="getDept" resultType="sample.vo.DeptVO">
    select * from DEPT
  </select>
  <delete id="deleteEmp" parameterType="sample.vo.EmpVO">
    delete from EMP where EMPNO=#{empno}
  </delete>
  <insert id="insertEmp" parameterType="sample.vo.EmpVO">
    insert into EMP(EMPNO, ENAME, SAL, DEPTNO)
    values("#{empno}", #{ename}, #{sal}, #{deptno})
  </insert>
  <update id="updateEmp" parameterType="sample.vo.EmpVO">
    update EMP
    set SAL=#{sal}, ENAME=#{ename}
    where EMPNO=#{empno}
  </update>
</mapper>

```

select 구문은 MyBatis 에서 가장 흔히 사용할 요소로 데이터베이스에서 데이터를 가져오는 기능을 수행합니다. 아마도 대부분의 애플리케이션은 데이터를 수정하기보다는 조회하는 기능을 많이 가집니다. 그래서 MyBatis 는 데이터를 조회하고 그 결과를 매핑하는데 집중하고 있습니다.

select 는 다음 예처럼 단순한 경우에는 단순히 설정됩니다.

```

<select id="selectPerson" parameterType="int" resultType="hashmap">
  SELECT * FROM PERSON WHERE ID = #{id}
</select>

```

이 구문의 이름은 selectPerson 이고 int 타입의 파라미터를 가집니다. 그리고 결과 데이터는 HashMap 에 저장됩니다. 파라미터 표기법은 다음과 같습니다.

```
#{id}
```

이 표기법은 MyBatis 에게 PreparedStatement 파라미터를 만들도록 지시합니다. JDBC 를 사용할 때 PreparedStatement 에는 “?” 형태로 파라미터가 전달됩니다. 즉 결과적으로 위 설정은 아래와 같이 작동하게 되는 셈입니다.

```

String selectPerson = "SELECT * FROM PERSON WHERE ID=?";
PreparedStatement ps = conn.prepareStatement(selectPerson);
ps.setInt(1,id);

```

데이터를 변경하는 구문인 insert, update, delete 는 Mapper XML 예제의 같이 매우 간단합니다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

6.3.2 Parameters/Result Maps

First Class(첫번째 요소) 인 select, insert, update, delete 는 parameterType 속성을 갖고 select 는 resultType 속성도 갖습니다. 이러한 속성의 값은 패키지 경로를 포함한 전체 클래스명이 사용됩니다.

Mapper XML 예제에서는 sample.vo.EmpVO 와 sample.vo.DeptVO 를 보았을 것이며, 이러한 클래스는 자바빈 형태의 VO 클래스로써, 프로퍼티와 getter setter 메소드로 구성됩니다.

다음은 sample.vo.DeptVO 예입니다.

```
package sample.vo;
public class DeptVO {
    private String deptno;
    private String dname;
    private String loc;
    public String getDeptno() {
        return deptno;
    }
    public void setDeptno( String deptno ) {
        this.deptno = deptno;
    }
    public String getDname() {
        return dname;
    }
    public void setDname( String dname ) {
        this.dname = dname;
    }
    public String getLoc() {
        return loc;
    }
    public void setLoc( String loc ) {
        this.loc = loc;
    }
}
```

다음은 sample.vo.EmpVO 예입니다.

```
package sample.vo;
import java.util.Date;
public class EmpVO {
    private Long empno;
    private String ename;
    private String job;
    private Long mgr;
    private Date hiredate;
    private Long sal;
    private Long comm;
    private Long deptno;
    public Long getEmpno() {
        return empno;
    }
    public void setEmpno( Long empno ) {
        this.empno = empno;
    }
}
```

```

public String getName() {
    return ename;
}
public void setName( String ename ) {
    this.ename = ename;
}
public String getJob() {
    return job;
}
public void setJob( String job ) {
    this.job = job;
}
public Long getMgr() {
    return mgr;
}
public void setMgr( Long mgr ) {
    this.mgr = mgr;
}
public Date getHiredate() {
    return hiredate;
}
public void setHiredate( Date hiredate ) {
    this.hiredate = hiredate;
}
public Long getSal() {
    return sal;
}
public void setSal( Long sal ) {
    this.sal = sal;
}
public Long getComm() {
    return comm;
}
public void setComm( Long comm ) {
    this.comm = comm;
}
public Long getDeptno() {
    return deptno;
}
public void setDeptno( Long deptno ) {
    this.deptno = deptno;
}
}

```

6.3.3 GlueParameter

GlueMyBatisDao 의 실행하고자 하는 Mapped SQL Statement 가 input parameter 값을 필요로 한다면, GlueParameter 를 통해 그 값을 전달합니다. select, insert, update, delete 구분에 parameterType 를 근거로 GlueParameter 를 생성합니다.

Mapped SQL Statement 가 다음과 같다면,

```
<select id="getEmp" parameterType="String" resultType="sample.vo.EmpVO">
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
select * from EMP where DEPTNO=#{deptno}
</select>
```

GlueParameter 는 다음과 같이 생성할 수 있습니다.

```
GlueParameter<String> param = new GlueParameter<String>();
param.setParameter( "10" );
```

Mapper SQL Statement 의 parameterType 이 다음과 같다면,

```
<select id="getEmp3" parameterType="sample.vo.EmpVO" resultType="sample.vo.EmpVO">
    select * from EMP where DEPTNO=#{deptno} and sal>#{sal}
</select>
```

GlueParameter 는 다음과 같이 생성할 수 있습니다. EmpVO 는 [Parameters/Reult Maps](#) 장의 예제와 같습니다.

```
GlueParameter<EmpVO> param = new GlueParameter<EmpVO>();
EmpVO empvo = new EmpVO();
empvo.setDeptno( (long) 10 );
empvo.setSal( (long) 10 );
param.setParameter( empvo );
```

6.3.4 Reuse & Custom Activity

GlueMybatisDao 를 사용한 reuse Activity 에는 GlueMybatisSearch, GlueMybatisDelete, GlueMybatisInsert, GlueMybatisModify Activity 가 있습니다.

6.3.4.1 GlueMybatisSearch

GlueMybatisDao 를 이용하여 Select 기능을 수행합니다.

```
<activity name="조회[Search]"
class="com.poscoict.glueframework.biz.activity.mybatis.GlueMybatisSearch">
    <transition name="success" value="end" />
    <property name="sql-key" value="emp.mapper.getEmp" />
    <property name="result-key" value="empList" />
    <property name="dao" value="dao-1" />
    <property name="param-bindings" value="deptno=DeptnoP" />
</activity>
```

- dao : (필수) applicationContext.xml 의 DAO id
- sql-key : (필수) Mapper XML 의 select id
- param-bindings : (선택) binding 에 사용되는 값과 Mapping 되는 Key
- result-key : (선택) Context 에 담기는 Query 수행 결과 Key.
- is-audit : (선택) GlueParameter 를 생성할때 GlueContext 에 auditAttributes 가 설정되어 있다면 해당 값을 GlueParameter 에 설정(자세한 내용은 16. Audit 참고).

6.3.4.2 GlueMybatisInsert

GlueMybatisDao 를 이용하여 Insert 기능을 수행합니다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
<activity name="등록[Insert]"
    class="com.poscoict.glueframework.biz.activity.mybatis.GlueMybatisInsert">
    <transition name="success" value="조회[Search]" />
    <property name="sql-key" value="emp.mapper.insertEmp" />
    <property name="dao" value="dao-1" />
    <property name="param-bindings"
        value="empno=EMPNO|ename=ENAME|sal=SAL|deptno=DEPTNO" />
    <property name="result-key" value="insertCnt" />
</activity>
```

- dao : (필수) applicationContext.xml 의 DAO id.
- sql-key : (필수) Mapper XML 의 insert id
- param-bindings : (선택) binding 에 사용되는 값과 Mapping 되는 Key
- result-key : (선택) Context 에 담기는 Query 수행 결과 Key.
- chk-name : (선택) 화면의 CheckBox ID - <property name="chk-name" value="chk" /> 와 같이 "chk" 가 선언되어 있다면 해당 값에 따라 Looping 처리를 한다.
- list-key : (선택) SQL 문에 Binding 할 Data - Context Key <property name="list-key" value="dataList" /> 와 같이 "dataList" 가 선언 되어 있다면 이전 Activity 에서 List(Map<name, value>) 구조의 Data 를 생성해서 Context 에 담는다.
- is-audit : (선택) true 이면 GlueParameter 를 생성할때 GlueContext 에 auditAttributes 가 설정되어 있다면 해당 값을 GlueParameter 에 설정(자세한 내용은 16. Audit 참고).

6.3.4.3 GlueMybatisModify

GlueMybatisDao 를 이용하여 Update 기능을 수행합니다.

```
<activity name="수정[Modify]"
    class="com.poscoict.glueframework.biz.activity.mybatis.GlueMybatisModify">
    <transition name="success" value="조회[Search]" />
    <property name="sql-key" value="emp.mapper.updateEmp" />
    <property name="dao" value="dao-1" />
    <property name="param-bindings" value="empno=EMPNO|ename=ENAME|sal=SAL" />
    <property name="result-key" value="updateCnt" />
</activity>
```

- dao : (필수) applicationContext.xml 의 DAO id.
- sql-key : (필수) Mapper XML 의 update id
- param-bindings : (선택) binding 에 사용되는 값과 Mapping 되는 Key
- result-key : (선택) Context 에 담기는 Query 수행 결과 Key
- chk-name : (선택) 화면의 CheckBox ID - <property name="chk-name" value="chk" /> 와 같이 "chk" 가 선언되어 있다면 해당 값에 따라 Looping 처리를 한다.
- list-key : (선택) SQL 문에 Binding 할 Data - Context Key <property name="list-key" value="dataList" /> 와 같이 "dataList" 가 선언 되어 있다면 이전 Activity 에서 List(Map<name, value>) 구조의 Data 를 생성해서 Context 에 담는다
- is-audit : (선택) true 이면 GlueParameter 를 생성할때 GlueContext 에 auditAttributes 가 설정되어 있다면 해당 값을 GlueParameter 에 설정(자세한 내용은 16. Audit 참고).

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

6.3.4.4 GlueMybatisDelete

GlueMybatisDao 를 이용하여 Delete 기능을 수행합니다.

```
<activity name="삭제[Delete]"
    class="com.poscoict.glueframework.biz.activity.mybatis.GlueMybatisDelete">
    <transition name="success" value="조회[Search]" />
    <property name="sql-key" value="emp.mapper.deleteEmp" />
    <property name="dao" value="dao-1" />
    <property name="param-bindings" value="empno=EMPNO" />
    <property name="result-key" value="deleteCnt" />
</activity>
```

- dao : (필수) applicationContext.xml 의 DAO id.
- sql-key : (필수) Mapper XML 의 delete id
- param-bindings : (선택) binding 에 사용되는 값과 Mapping 되는 Key
- result-key : (선택) Context 에 담기는 Query 수행 결과 Key
- chk-name : (선택) 화면의 CheckBox ID - <property name="chk-name" value="chk" /> 와 같이 "chk" 가 선언되어 있다면 해당 값에 따라 Looping 처리를 한다.
- list-key : (선택) SQL 문에 Binding 할 Data - Context Key <property name="list-key" value="dataList" /> 와 같이 "dataList" 가 선언 되어 있다면 이전 Activity 에서 List(Map<name, value>) 구조의 Data 를 생성해서 Context 에 담는다

6.3.4.5 Custom Activity

Reuse Activity 를 사용하지 않고 applicationContext.xml 에 정의된 DAO 빈을 이용하여 Data CRUD 를 할수 있습니다.

다음은 Custom Activity 일부입니다.

```
GlueGenericDao dao = this.getDao("dao-1");

List<DeptVo> result = dao.find("emp.mapper.getDept");
for( int i=0; i<result.size(); i++ ){
    DeptVo dept = result.get( i );
    logger.logInfo( dept.getDeptno() );
}

GlueParameter<EmpVo> param = new GlueParameter<EmpVo>();
EmpVO emp = new EmpVO();
emp.setEmpno( (long) 9999 );
emp.setName( "JIN" );
emp.setDeptno( (long) 10 );
emp.setSal( (long) 10 );
param.setParameter( emp );

int intCnt = dao.update( "emp.mapper.insertEmp", param );

emp.setSal( (long) 100 );
int updateCnt = dao.update( "emp.mapper.updateEmp", param );
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
int deleteCnt = dao.delete( "emp.mapper.deleteEmp", param );
```

6.4 GlueHibernateDao

GlueHibernateDao 는 SpringJDBC 를 이용해서 추상화된 CRUD 오퍼레이션을 제공하는 구현체입니다. GlueJdbcDao 는 GlueGenericJdbcDao 인터페이스를 구현하였으며, GlueGenericJdbcDao 는 GlueGenericDao 를 확장하였으며, 그 메소드는 다음과 같습니다

GlueHibernateDao 는 Hibernate 를 이용해서 추상화된 CRUD 오퍼레이션을 사용할 수 있는 Database Access Object 구현체입니다. GlueHiberanteDao 는 GlueGenericHibernateDao 인터페이스를 구현하였으며, GlueGenericHibernateDao 는 GlueGenericDao 를 확장하였으며, 그 메소드는 다음과 같습니다.

method	설명
find(String queryKey) find(String queryKey, GlueParameter param)	xxx-hquery.glue_sql 에 정의한 Query 로 DB 레코드를 조회/수정/삽입/삭제한다.
update(String queryKey) update(String queryKey, GlueParameter param)	
delete(String queryKey) delete(String queryKey, GlueParameter param)	
findByHql(String queryStmnt) findByHql(String queryStmnt, GlueParameter param)	프로그램 소스상에서 Query 를 직접 작성하여 DB 레코드를 조회/수정/삽입/삭제한다.
updateByHql(String queryStmnt) updateByHql(String queryStmnt, GlueParameter param)	
deleteByHql(String queryStmnt) deleteByHql(String queryStmnt, GlueParameter param)	
createEntity(Object entity) createEntity(String entityName, GlueParameter param) loadEntity(String entityName, Serializable id) modifyEntity(String entityName, Object entity, GlueParameter param) removeEntity(Object entity) removeEntity(String entityName, Serializable entityId)	HQL 을 사용하지 않고 VO(entity)를 생성/등록/수정/로딩한다.

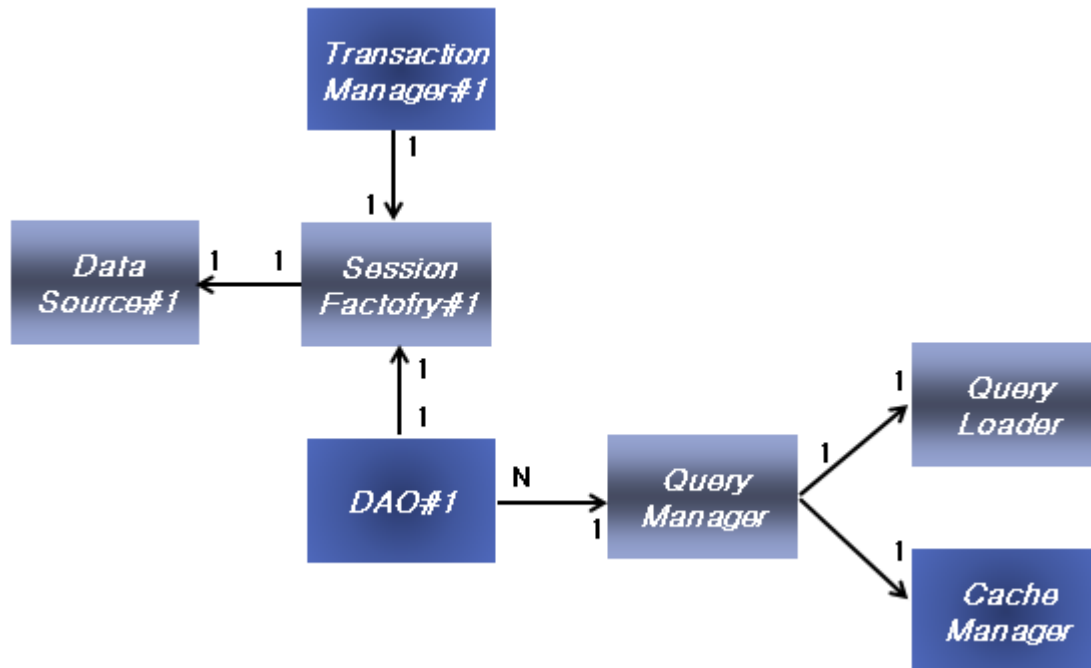
[표 12] GlueGenericHibernateDao 메소드

본 가이드에서는 Glue 와 Hibernate 연계 관련된 기본적인 내용에 대해서만 다룰 것이므로, Hibernate 에 대한 자세한 내용은 Hibernate 공식 사이트의 매뉴얼을 참고하기 바랍니다.

영문 사이트 : <http://www.hibernate.org/>

GlueHibernateDao 를 사용하기 위해서는 applicationContext.xml 에 SessionFactory, QueryManager 가 등록되어 있어야 합니다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 44] GlueHibernateDao 의 bean 들과의 관계

다음은 설정파일(applicationContext.xml)의 일부입니다.

```

<bean id="dataSource-1" class= . . ./>
<bean id="dataSource-2" class= . . ./>
<bean id="transactionManager-1"
    class="com.poscoict.glueframework.transaction.GlueHibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory-1"/>
</bean>
<bean id="transactionManager-2"
    class="com.poscoict.glueframework.transaction.GlueHibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory-2"/>
</bean>
<bean id="dao-1" class="com.poscoict.glueframework.dao.hibernate.GlueHibernateDao">
    <property name="sessionFactory" ref="sessionFactory-1"/>
    <property name="queryManager" ref="queryManager"/>
</bean>
<bean id="dao-2" class="com.poscoict.glueframework.dao.hibernate.GlueHibernateDao">
    <property name="sessionFactory" ref="sessionFactory-2"/>
    <property name="queryManager" ref="queryManager"/>
</bean>
<bean id="sessionFactory-1"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="dataSource" ref=" dataSource-1"/>
    <property name="mappingResources">
        <list>
            <value>HbSample01.hbm.xml</value>
        </list>
    </property>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">org.hibernate.dialect.HSQLDialect</prop>
            <prop key="hibernate.show_sql">true</prop>
        </props>
    </property>
</bean>
  
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

        <prop key="hibernate.transaction.flush_before_completion">false</prop>
    </props>
</property>
</bean>
<bean id="sessionFactory-2"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource-2"/>
    <property name="mappingResources">
        <list>
            <value>Sample01.hbm.xml</value>
        </list>
    </property>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">org.hibernate.dialect.HSQLDialect</prop>
            <prop key="hibernate.show_sql">true</prop>
            <prop key="hibernate.transaction.flush_before_completion">false</prop>
        </props>
    </property>
</bean>
<bean id="queryManager"
    class="com.poscoict.glueframework.dao.manager.GlueQueryManagerImpl">
    <property name="cacheManager" ref="cacheManager"/>
    <property name="queryLoader" ref="queryLoader"/>
</bean>
<bean id="queryLoader" class="com.poscoict.glueframework.dao.manager.GlueQueryLoader">
    <property name="rootPath" value="query"/>
</bean>
<bean id="cacheManager" class=". . ."/>

```

“dao-1”은 Hibernate 용 DAO bean 으로 class 는

“com.poscoict.glueframework.dao.hibernate.GlueHibernateDao”을 사용하며, SessionFactory

객체와 QueryManager 객체를 필요로 합니다. “dao-1”의 sessionFactory property 의 값인

“sessionFactory-1”은 Hibernate Session 관리 객체로 Hibernate 관련 설정 정보를 셋업합니다.

“sessionFactory-1”의 mappingResources property 에는 hbm.xml 을 설정하며, Database 와 연동되는 Value Object 정보가 포함되어 있습니다. hibernateProperties property 에는 Hibernate 정책 및 세부 설정 사항을 지정한합니다. Hibernate 설정 방법(sessionFactory-1 Bean 의 hibernateProperties)은 Hibernate 공식 사이트를 참고하기 바랍니다.

6.4.1 hbm.xml

Hibernate ORM 매핑방식 중 hbm.xml 매핑 설정 방식이 있습니다. hbm.xml 에 Database Schema 를 정의하고 Hibernate 의 설정을 지정하는 파일입니다. Hibernate 사용방법에 대해 정확한 습득을 필요로 하면 Hibernate 공식 사이트를 참고하시기 바랍니다.

다음은 hbm.xml 예제입니다.

```

<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<class name="sample.vo.Employee" table="Emp" dynamic-update="true">
  <id name="empno" column="EMPNO">
    <generator class="increment"/>
  </id>
  <property name="ename" column="ENAME" type="string"/>
  <property name="job" column="JOB" type="string"/>
  <property name="mgr" column="MGR" type="big_decimal"/>
  <property name="hiredate" column="HIREDATE" type="timestamp"/>
  <property name="sal" column="SAL" type="big_decimal"/>
  <property name="comm" column="COMM" type="big_decimal"/>
  <property name="deptno" column="DEPTNO" type="big_decimal"/>
</class>
</hibernate-mapping>

```

위와 같은 작성된 hbm.xml 을 살펴보면, class 요소의 name 을 통해 Value Object Class 가 지정되어 있고, Entity 를 구분하기 위한 id 가 있고, 그외 property 들을 볼 수 있습니다. Id 는 increment generator 를 사용하고 있습니다.

Value Object 는 DB 컬럼에 해당하는 멤버 변수와 Setter / Getter Method 를 갖습니다. 상황에 따라 Business Logic 이 들어갈 수 있으며 Hibernate 관련 Annotation 을 사용할 수 있습니다.

다음은 sample.vo.Employee 예제입니다. hbm.xml 의 id 로 지정된 empno 는 increment generator 를 사용하도록 하였기 때문에 long, short 과 같은 타입만 허용됩니다. 만약 다른 Type 을 사용하려면 generator 를 직접 개발해야 합니다.

```

package sample.vo;
import java.math.BigDecimal;
import java.sql.Timestamp;
public class Employee {
    // bhm.xml 에서 increment generator 를 사용하려면 long 형을 사용해야 한다.
    private long empno;
    private String ename;
    private String job;
    private BigDecimal mgr;
    private Timestamp hiredate;
    private BigDecimal sal;
    private BigDecimal comm;
    private BigDecimal deptno;
    public Employee( long empno ) {
        this.empno = empno;
    }
    public long getEmpno() {
        return empno;
    }
    public void setEmpno( long empno ) {
        this.empno = empno;
    }
    public String getEname() {
        return ename;
    }
    public void setName( String ename ) {
        this.ename = ename;
    }
    public String getJob() {
        return job;
    }
}

```

```

public void setJob( String job ) {
    this.job = job;
}
public BigDecimal getMgr() {
    return mgr;
}
public void setMgr( BigDecimal mgr ) {
    this.mgr = mgr;
}
public Timestamp getHiredate() {
    return hiredate;
}
public void setHiredate( Timestamp hiredate ) {
    this.hiredate = hiredate;
}
public BigDecimal getSal() {
    return sal;
}
public void setSal( BigDecimal sal ) {
    this.sal = sal;
}
public BigDecimal getComm() {
    return comm;
}
public void setComm( BigDecimal comm ) {
    this.comm = comm;
}
public BigDecimal getDeptno() {
    return deptno;
}
public void setDeptno( BigDecimal deptno ) {
    this.deptno = deptno;
}
}

```

다음은 EMP table 생성 Script 예제입니다. 아래 예제는 Oracle 용 EMP 생성 Script 입니다.

```

CREATE TABLE EMP
(
    EMPNO      NUMBER (4) NOT NULL,
    ENAME      VARCHAR2 (10),
    JOB        VARCHAR2 (9),
    MGR        NUMBER (4),
    HIREDATE    DATE,
    SAL        NUMBER (7,2),
    COMM       NUMBER (7,2),
    DEPTNO     NUMBER (2),
    CONSTRAINT FK_DEPTNO FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO)
)

```

6.4.2 Glue Query File(HQL)

O/R 맵핑 툴의 하나인 Hibernate 는 HQL 에 의한 질의를 수행합니다. HQL 이란 Hibernate Query Language 의 약어로 Hibernate 를 통해 데이터베이스를 검색할 경우에 이용하는 쿼리입니다. HQL 을

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Hibernate 가 어떤 데이터베이스를 이용하고 있는지 살펴보고 알아서 그에 맞는 SQL 로 변환하여 준다는 장점이 있습니다. HQL 은 SQL 에 비슷하지만 클래스의 프로퍼티를 쓴다는 점이 특징입니다.

Glue Query File 작성시 JDBC 용 Query 와 Hibernate HQL 를 구분하기 위해 [-hquery.glue_sql]과 같이 Query File 명을 지정합니다. 작성되는 HQL 은 named query 형태로만 작성합니다. 그리고 isNamed 속성을 true 로 합니다. Hibernate 를 사용하기 위해서는 일반적인 DB 벤더에서 사용하는 SQL 과 상이한 HQL 언어에 대해서 알고 있어야 합니다. 다음은 [hbm.xml](#) 에서 예제로 사용한 EMP 테이블을 매핑한 Employee 를 대상으로 한 HQL 로 작성된 Glue Query File 일부입니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<queryMap desc="Employee Query for Hibernate">
  <query id="Employee.find" desc="find employee by empno" isNamed="true">
    <![CDATA[
from Employee e where e.empno = :empno order by e.empno asc
    ]]>
  </query>
  <query id="Employee.findByDept" desc="find employee by detpno" isNamed="true">
    <![CDATA[
from Employee e where e.deptno = :deptno order by e.empno asc
    ]]>
  </query>
  <query id="Employee.update" desc="find employee by detpno" isNamed="true">
    <![CDATA[
update Employee set comm = :comm where empno = :empno
    ]]>
  </query>
  <query id="Employee.delete" desc="find employee by detpno" isNamed="true">
    <![CDATA[
delete Employee where empno = :empno
    ]]>
  </query>
</queryMap>
```

“Employee.find”라는 query id 로 부여된 HQL 은 Emp Table 에서 binding 한 deptno 와 동일한 사원을 조회하여 Employee 객체의 List 를 Return 하는 쿼리입니다. from 뒤의 Employee 는 DB Table 명이 아닌 Hibernate 의 Entity 명임에 주의합니다. HQL 에서는 named query 와 ? 방식의 query 도 지원하나, Glue Framework 의 Reusable Activity 는 named Query 만 지원하고 있습니다.

“Employee.update”라는 query id 로 부여된 HQL 은 Emp Table 에서 binding 한 사원번호의 Comm 컬럼을 Update 하는 쿼리입니다.

“Employee.delete”라는 query id 로 부여된 HQL 은 Emp Table 에서 사원번호로 Delete 하는 쿼리입니다.

Hibernate 에서 Insert 쿼리는 제한적으로 지원하나, Glue Framework 에서는 제공하는 Resue Activity 에는 insert 쿼리를 위한 것은 없습니다. Hibernate 에서 제한적으로 insert HQL 이 지원되기에 Glue Framework 에서는 Entity 기반의 Reuse Activity 를 이용하기를 권고합니다.

Hibernate 에서는 HQL 을 통해 기본적인 Insert, Update, Delete 구문을 지원하지만, GlueHibernateDao 를 통해 사용하지 않을 것을 권고합니다. Hiberante 를 이용하여 Data 를 조회한 후 Custom Activity 등에서 각 Value Object 에 접근하여 Business Data 를 변경한 후 Commit 을 통해 Database 에 변경된 내용을 적용하는 방식으로 개발하기 바랍니다. insert/update/delete HQL 의 사용은 Hibernate 의 OR mapping 사상과 맞지 않는 개념이기 때문입니다. Value Object 을 통한 CRUD 가 아닌 경우는 GlueJdbcDao/GlueMyBatisDao 의 Data 수정/등록/삭제 기능을 활용 할 수 있습니다.

아래는 Hibernate VO 에 Update 하는 Sample Code 입니다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

public class SampleActivity extends GlueActivity {
    public String runActivity(GlueContext ctx) {
        GlueGenericHibernateDao dao = . . .;
        GlueParameter param = . . .;
        List<Employee> rowSet = dao.find("Employee.findByDept", param);
        for ( Employee emp : voList ) {
            // Logic 을 작성한다
            if( isUpdateEmployee(emp) ) {
                emp.setComm(new BigDecimal("1"));
            }
        }
        // 필요시 commit/rollback 을 activity layer 에서 처리 가능함.
        // this.commitTransaction( "transactionManager-1" );
        // this.rollbackTransaction( "transactionManager-1" );
        return "success";
    }
    private boolean isUpdateEmployee(Employee emp){
        . . . .
    }
}

```

6.4.3 GlueHiberanteParameter

GlueHibernateDao 를 이용한 HQL based/Entity Based method 실행시 GlueParameter 를 필요로 합니다. HQL 은 Named Query 만을 사용할 수 있으므로, GlueParameter 는 Map 으로 구성합니다. 구성된 Map 객체는 setParameter() 를 이용하거나, 생성자를 이용합니다.

GlueHiberanteParameter 는 entityName 과 bindingNamings 를 추가로 갖고 있습니다. entityName 은 update()/delete()/createEntity()/modifyEntity() 메소드 사용시 date 의 TypeCasting 에 사용됩니다. bindingNamings 는 find()/update()/delete() 와 createEntity()/modifyEntity() 메소드 사용시 entity 의 Field 명과 dataMap 의 key 가 일치하지 않을 경우 사용됩니다.

다음은 find() 메소드를 사용할 경우 GlueParameter 를 구성하는 예입니다.

```

Map<String, String> dataMap = new HashMap<String, String>();
dataMap.put( "deptno", "40" );
GlueParameter param = new GlueHibernateParameter<Map<String, String>>( dataMap );
List<?> rowSet = dao.find( "Employee.findByDept", param );

```

다음은 named param 이 같지 않을 경우 GlueParameter 를 구성하는 예입니다.

```

// from Employee e where e.deptno = :deptno1 order by e.empno asc
Map<String, String> dataMap = new HashMap<String, String>();
dataMap.put( "deptno1", "40" );
Map<String, String> bindings = new HashMap<String, String>();
bindings.put( "deptno1", "deptno" );
GlueParameter param = new GlueHibernateParameter<Map<String, String>>();
param.setParameter( dataMap );
( (GlueHibernateParameter<?>) param ).setBindingNamings( bindings );
List<?> rowSet = dao.find( "Employee.findByDept.2", param );

```

다음은 update/delete() 메소드를 사용할 경우 GlueParamter 를 구성하는 예입니다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
Map<String, String> dataMap = new HashMap<String, String>();
dataMap.put( "empno", "1111" );
dataMap.put( "comm", "3333" );
GlueParameter param = new GlueHibernateParameter<Map<String, String>>( dataMap );
( (GlueHibernateParameter<?>) param ).setEntityName( "sample.vo.Employee" );
int cnt = dao.update( "Employee.update", param );
```

다음은 createEntity()/modifyEntity() 메소드를 사용할 경우 GlueParameter 를 구성하는 예입니다.

```
Map<String, String> dataMap = new HashMap<String, String>();
dataMap.put( "empno", "4111" );
dataMap.put( "ename", "Test1" );
dataMap.put( "deptno", "40" );
GlueParameter param = new GlueHibernateParameter<Map<String, String>>( dataMap );
( (GlueHibernateParameter<?>) param ).setEntityName( "sample.vo.Employee" );
Object obj = dao.createEntity( "sample.vo.Employee", param );
```

6.4.4 Reuse & Custom Activity

Hibernate Activity 의 목적은 POJO 기반의 코딩을 지원하기 위한 것으로 Mybatis 또한 Value Object 를 지원합니다.

Hibernate 를 위한 Reusable Activity 크게 Query Based Activity 와 Entity Baseed Activity 두 부류로 분류됩니다. Query Based Activity 는 Hibernate 의 HQL(Hibernate Query Language) 기반의 Action 을 할 수 있도록 지원합니다. Hibernate 의 장점을 살려 프로그램을 작성하기 위해서 Update, Delete, Insert 의 경우 Custom Activity 또는 Entity 기반의 Activity 를 활용하는 것이 좋습니다.

Query Based Activity 는 JDBC 방식의 Activity 사용방법과 동일하게 작성된 Query File 의 HQL 을 사용합니다. Query Based Activity 에서 사용하는 HQL 은 현재 Named Query 만 지원하며 HQL 의 Named 변수에 Binding 하는 값은 GlueContext 에서 Named 변수 명으로 Get 하여 Binding 한다.

Query Based Reuse Activity 에는 GlueHibernateSearch, GlueHibernateUpdate, GlueHibernateDelete 가 있습니다. Query Based Reuse Activity 를 통해서 Data insert 는 제공하지 않습니다.

Entity Based Activity 를 통해서 Hibernate 의 Entity 명과 Entity Id 값을 이용하여 객체를 찾고, 등록, 수정, 삭제할 수 있습니다. Entry Based Reuse Activity 에는 GlueHibernateEntityCreator, GlueHibernateEntityModifier, GlueHibernateEntityRemover 가 있습니다.

Glue 에서 지원하는 Activity 는 Hibernate 의 일부 기능을 호출해주는 역할을 수행하며 실질적인 조회, 수정, 삭제, 등록의 로직은 Hibernate 내부에서 일어난다. 따라서 Hibernate 의 내부적으로 수행되는 원리와 설정 그리고 HQL 을 알지 못할 경우 성능에 큰 문제가 발생할 수 있기 때문에 Hibernate 를 잘 아는 개발자가 사용할 것을 권고한다.

6.4.4.1 GlueHibernateSearch

Select 용 HQL 을 수행하여 Value Object 의 List 형태로 결과를 저장하는 Activity 입니다. 추가적으로 Value Object 의 특정 Method 를 실행시키는 기능도 포함합니다. Business Logic 을 Custom Activity 에서 관리할 것인지 Entity(Value Object)에서 관리할지에 대한 고민이 생각하고 적절하게 판단할 필요가 있다

```
<activity name="조회"
class="com.poscoict.glueframework.biz.activity.hibernate.GlueHibernateSearch">
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<transition name="success" value="end" />
<property name="result-key" value="empList" />
<property name="sql-key" value="Employee.findByDept" />
<property name="dao" value="dao-1" />
</activity>

```

- dao : hibernate dao 명
- sql-key : HQL query 명
- result-key : HQL 수행 결과를 GlueContext 에 저장할 명. 수행 결과는 HQL 에 지정된 Object 의 List 이다.
- method-name : 조회 후 Value Object 에서 실행할 Method 명
- method-param0 : Method 에 넘겨줄 값

6.4.4.2 GlueHibernateUpdate

Update 용 HQL 을 수행하여 DB 에 변경사항을 반영하고 Update 된 수를 Return 하는 Activity 입니다.

```

<activity name="수정"
class="com.poscoict.glueframework.biz.activity.hibernate.GlueHibernateUpdate">
    <transition name="success" value="end" />
    <property name="result-key" value="updateCount" />
    <property name="sql-key" value="Employee.update" />
    <property name="entity-name" value="sample.vo.Employee" />
    <property name="dao" value="dao-1" />
</activity>

```

- dao : hibernate dao 명
- sql-key : HQL query 명
- result-key : HQL 수행 결과를 GlueContext 에 저장할 명. 수행 결과는 HQL 에 지정된 Object 의 List 이다.

6.4.4.3 GlueHibernateDelete

Update 용 HQL 을 수행하여 DB 에 Data 를 삭제하고 Delete 된 수를 Return 하는 Activity 이다.

```

<activity name="삭제"
class="com.poscoict.glueframework.biz.activity.hibernate.GlueHibernateUpdate">
    <transition name="success" value="end" />
    <property name="result-key" value="deleteCount" />
    <property name="sql-key" value="Employee.delete" />
    <property name="entity-name" value="sample.vo.Employee" />
    <property name="dao" value="dao-1" />
</activity>

```

- dao : hibernate dao 명
- sql-key : HQL query 명
- result-key : HQL 수행 결과를 GlueContext 에 저장할 명. 수행 결과는 삭제된 Object 의 수이다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

6.4.4.4 GlueHibernateEntityCreator

이 Activity 는 hbm.xml 에 정의된 Entity 을 생성하는 Activity 이다. 생성된 ValueObject 의 멤버변수 명으로 GlueContext 에서 값을 가져와서 세팅한다.

```
<activity name="생성"
class="com.poscoict.glueframework.biz.activity.hibernate.GlueHibernateEntityCreator">
    <transition name="success" value="생성 2" />
    <property name="dao" value="test-dao" />
    <property name="entity-name" value="sample.vo.Employee" />
    <property name="result-key" value="id" />
</activity>
```

- dao : hibernate dao 명
- result-key : Load 한 Entity 를 GlueContext 에 저장할 명.
- entity-name : 가져올 Entity 의 이름

6.4.4.5 GlueHibernateEntityModifier

이 Activity 는 hbm.xml 에 정의된 Entity 을 수정하는 Activity 이다. 생성된 ValueObject 의 멤버변수 명으로 GlueContext 에서 값을 가져와서 수정한다.

```
<activity name="변경"
class="com.poscoict.glueframework.biz.activity.hibernate.GlueHibernateEntityModifier">
    <transition name="success" value="변경 2" />
    <property name="dao" value="test-dao" />
    <property name="entity-name" value="sample.vo.Employee" />
    <property name="entity-id" value="empno1" />
    <property name="result-key" value="emp" />
</activity>
```

- dao : hibernate dao 명
- result-key : Load 한 Entity 를 GlueContext 에 저장할 명.
- entity-name : 가져올 Entity 의 이름
- entity-id : Entity 를 구분할 Key 의 Id

6.4.4.6 GlueHibernateEntityRemover

이 Activity 는 hbm.xml 에 정의된 Entity 을 삭제하는 Activity 이다.

```
<activity name="제거"
class="com.poscoict.glueframework.biz.activity.hibernate.GlueHibernateEntityRemover">
    <transition name="success" value="조회 1" />
    <property name="result-key" value="cnt" />
    <property name="entity-name" value="sample.vo.Employee" />
    <property name="entity-id" value="empno" />
    <property name="dao" value="test-dao" />
</activity>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

</activity>

- dao : hibernate dao 명
- result-key : Load 한 Entity 를 GlueContext 에 저장할 명.
- entity-name : 삭제할 Entity 의 이름
- entity-id : Entity 를 구분할 Key 의 Id

6.4.4.7 Custom Activity

Reuse Activity 를 사용하지 않고 applicationContext.xml 에 정의된 DAO 빈을 이용하여 Data CRUD 를 할수 있습니다.

다음은 Custom Activity 일부입니다.

```
GlueGenericHibernateDao dao = (GlueGenericHibernateDao)this.getDao("dao-1");

//조회
List<?> result = dao.findByHql("from Employee where deptno=40");
for( int i=0; i<result.size(); i++ ){
    Employee emp = (Employee)result.get( i );
    logger.logInfo( dept.getDeptno() );
}

//생성
Employee emp = new Employee();
emp.setDeptno( new BigDecimal( "40" ) );
emp.setEmpno( 4444 );
emp.setEname( "Test1" );
Object obj = dao.createEntity( emp );

Map(String,String) dataMap = new HashMap<String, String>();
dataMap.put( "empno", "4445" );
dataMap.put( "ename", "Test2" );
dataMap.put( "deptno", "40" );
GlueParameter param = new GlueHibernateParameter<Map<String, String>>( dataMap );
( (GlueHibernateParameter<?>) param ).setEntityName( "sample.vo.Employee" );
obj = dao.createEntity( "sample.vo.Employee", param );

//수정
emp = (Employee) dao.loadEntity( "sample.vo.Employee", new Long( "4444" ) );
emp.setComm(5000);

dataMap = new HashMap<String, String>();
dataMap.put( "empno", "4445" );
dataMap.put( "comm", "6000" );
GlueParameter param = new GlueHibernateParameter<Map<String, String>>( dataMap );
( (GlueHibernateParameter<?>) param ).setEntityName( "sample.vo.Employee" );
int cnt = dao.update( "Employee.update", param );

//삭제
obj = dao.removeEntity( "sample.vo.Employee", new Long( "4444" ) );
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
dataMap = new HashMap<String, String>();
dataMap.put( "empno", "4445" );
GlueParameter param = new GlueHibernateParameter<Map<String, String>>( dataMap );
( (GlueHibernateParameter<?>) param ).setEntityName( "sample.vo.Employee" );
cnt = dao.update( "Employee.delete", param )
```

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

7. Cache 활용

7.1 정의 및 역할

본 문서에서 언급하는 Cache 라 함은 자바 오브젝트를 저장하는 메모리 상의 공간을 의미한다

JCS(Java Object Cache), EhCache 와 같은 것이 일종의 Cache 를 지원하는 것이라 생각할 수 있지만 단순히 오브젝트를 저장하고 서비스하는 것뿐만 아니라 분산 시스템 지원, 클러스터링 지원, 오브젝트/Region 별 만기일 지정, 오브젝트 모니터링, Cache Capacity 한정, 그룹 지정과 같은 여러 가지 부가 서비스를 가져야 비로서 Caching System 이라 할 수 있다.

7.2 사용 목적

오브젝트 생성에 코스트가 많이 소모되는 경우나 빈번하게 사용되는 Read-only 성의 오브젝트들을 Cache 에 저장하여 사용하는 방법을 권장한다.

Glue Framework 내부에서도 Service Definition, Message Definition, Query Definition 같은 Metadata 성격의 오브젝트를 Cache 에 등록하여 전체 애플리케이션의 성능을 높이고 있다. 따라서 각 개발자들도 이런 성격의 오브젝트는 분류하여 Cache 에 등록하여 사용하는 것이 바람직하다.

7.3 Caching Solution 소개 및 비교

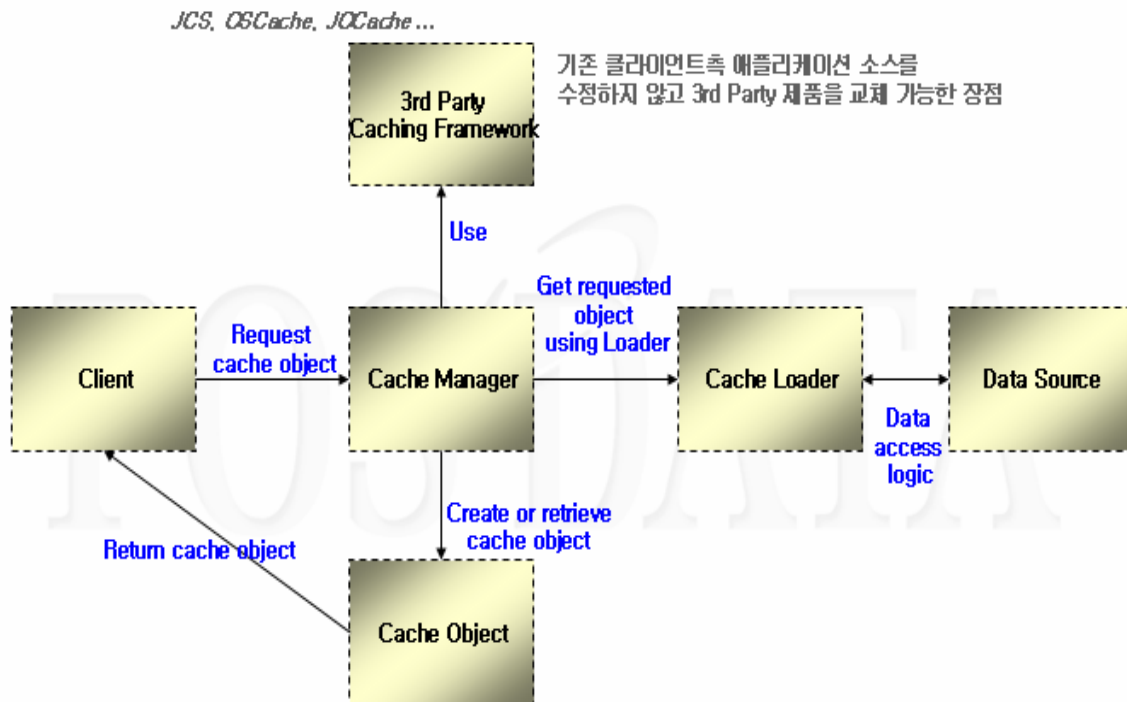
Open Source	Commercial
Java Caching System (JCS) OSCache Java Object Cache (JOCache) JBossCache EhCache	SpiritCache (SpiritSoft) Coherence (Tangosol) Object Caching Service for Java (Oracle)

[표 13] Caching solution

현존하는 Caching Solution 을 크게 Open Source 진영과 Commercial 두 종류로 볼 수 있다. 공통 인터페이스 추출 용이성 및 향후 솔루션 대체 시 변경 영향도를 최소화하기 위해서 JSR-127 에 호환되는 솔루션을 대상으로 비교 분석한 결과 Glue Framework 에서는 Apache 그룹에서 연구 했던 Java Caching System(JCS)을 도입하여 우리가 필요로 하는 기능을 추가한 Caching 컴포넌트를 제공하였으며 Glue 4.0 버전부터 EhCache 를 지원한다.

7.4 Glue Caching Component Architecture Diagram

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 45] CacheManager

Client 는 Cache Manager 에게 필요한 오브젝트를 요구한다. 해당하는 오브젝트가 존재할 때는 Cache 영역에서 오브젝트를 획득하여 Client 에게 서비스한다. 존재하지 않으면 오브젝트를 직접 생성하거나 Cache Loader 를 통하여 Data Source 에 접근하여 오브젝트를 얻은 후 Client 에게 서비스한다.

Client 가 직접 3rd Party Caching Solution 을 직접 사용하지 않고 Cache Manager 라는 공통 인터페이스를 통해서 접근하기 때문에 향후 기존 Client 측 소스를 수정하지 않고도 다른 Caching Solution 으로 교체 가능한 장점을 지닌 Flexible Architecture 이다.

7.5 JCS Cache 개발 가이드

Cache Manager 를 사용하기 위해서는 두 가지 선행 작업이 필요하다. 첫 번째로

applicationContext.xml 설정 파일에 'cacheManager'란 아이디를 갖는 Bean 이 등록되어 있어야 한다.

```

<!-- Cache Manager 설정 -->
<bean id="cacheManager"
class="com.poscoict.glueframework.cache.GlueCacheManagerJCS"/>

```

두 번째는 cache.ccf 란 설정 파일이 클래스패스 루트에 있어야 한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
## cache.ccf 예시
# DEFAULT CACHE REGION

# sets the default aux value for any non configured caches
jcs.default=
jcs.default.cacheattributes=org.apache.jcs.engine.ComGlueiteCacheAttributes
jcs.default.cacheattributes.MaxObjects=1000000
jcs.default.cacheattributes.MemoryCacheName=org.apache.jcs.engine.memory.lru.LRUMemoryCache
jcs.default.elementattributes=org.apache.jcs.engine.ElementAttributes
jcs.default.elementattributes.IsEternal=false
jcs.default.elementattributes.MaxLifeSeconds=-1
jcs.default.elementattributes.IdleTime=1
jcs.default.elementattributes.IsSpool=true
jcs.default.elementattributes.IsRemote=false
jcs.default.elementattributes.IsLateral=true
```

예시에서 제시하는 프로퍼티를 그대로 사용해도 무방하다. 각각의 프로퍼티의 자세한 의미는 JCS Configuration 을 참조하면 된다.

두 가지 작업을 완료하면 자바 코드에서 바로 Glue Caching Component 를 사용할 수 있다.

다음 예제 소스는 Glue Framework 에 적용한 Cache Region 을 사용하는 예제이다. Cache 에서 Region 이란 하나의 커다란 Cache 를 논리적으로 분할하는 것을 뜻한다. 여러 개의 Region 을 정의 할 수 있고 각각의 Region 은 서로 독립적이다. 즉 같은 키를 갖는 Cached Object 가 서로 다른 Region 에 존재할 수 있다는 것이다. Cache 에 저장하는 오브젝트의 성격에 따라 Region 을 분할해서 사용하는 것을 권장한다.

<cache loader 를 사용하지 않는 예제>

```
Integer value = (Integer)cacheManager.getCacheObject("key1", "IntegerRegion");
if (value == null)
{
    cacheManager.putCacheObject("key1", new Integer(6933), "IntegerRegion");
}
```

region 명이 "IntegerRegion"인 영역에서 cache key 가 "key1"에 해당하는 Integer Object 를 얻는다.

<cache loader 를 사용하는 예제>

```
Integer value2 =
    (Integer)cacheManager.getCacheObject("key1",
                                          "IntegerRegion",
                                          new MyIntegerLoader());
```

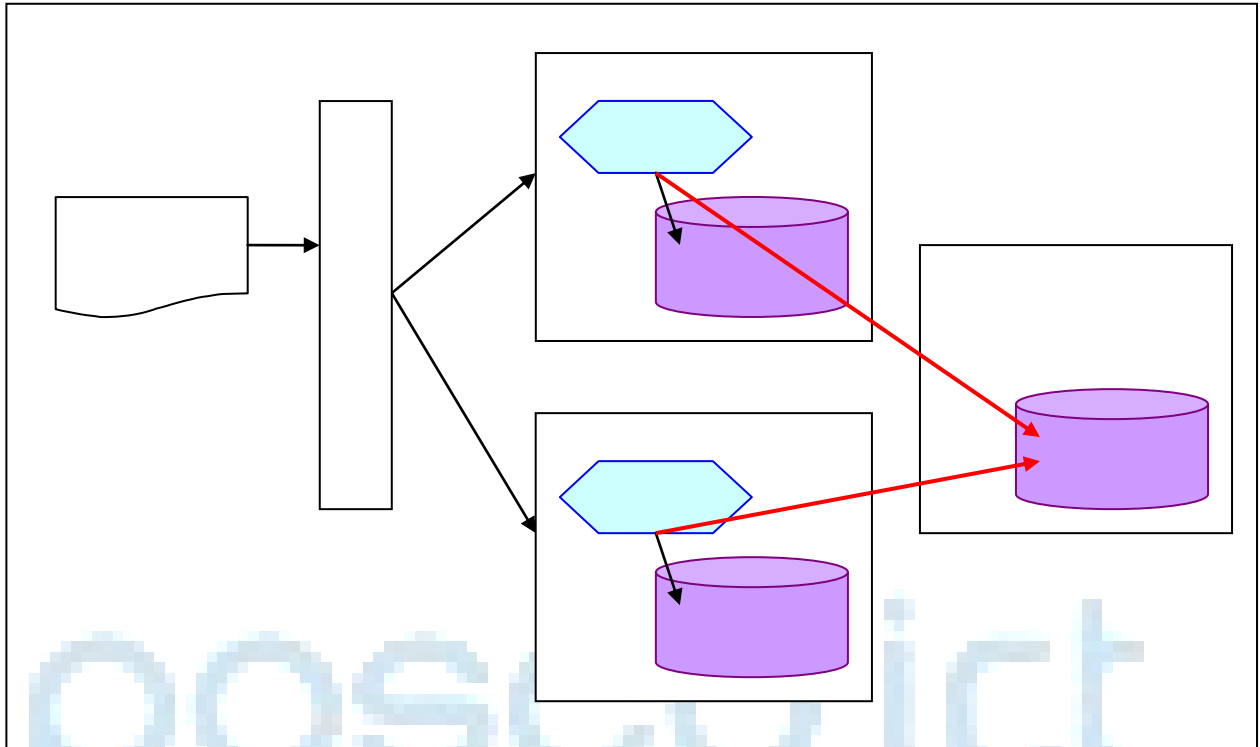
MyIntegerLoader 는 com.poscoict.glueframework.cache.GlueCacheLoader 인터페이스를 구현한 클래스이다. CacheManager 는 내부적으로 우선 "IntegerRegion" 영역에서 "key1"에 해당하는 오브젝트를 찾는다. 해당 오브젝트가 없으면 MyIntegerLoader.loadCacheObject 메소드가 실행되어 필요한 오브젝트를 로딩해 온다. MyIntegerLoader 오브젝트를 new 생성자를 이용해서 매번 생성하는 것으로 예제로 제시했지만 되도록이면 클래스 프로퍼티 (클래스 멤버 필드)로 지정하여 사용하는 것이 바람직하다. (불필요하게 과도한 인스턴스 생성 억제)

두 종류의 구현 방법 중에 취향에 맞는 방법을 선택하여 사용하기 바란다. 되도록이면 GlueCacheLoader 인터페이스를 구현한 MyIntegerLoader 와 같이 Custom Loader 클래스를 사용하는 것이 유지보수성과 확장성 측면에서 유리하기 때문에 적극 권장하는 방법이다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

7.5.1 Remote Cache

Application Server 가 클러스터링 되어 있고, 프로그램에서 사용되는 공통 Data 에 대한 관리를 DB 가 아닌 메모리에 하고자 한다면 다음과 같이 시스템을 구성할 수 있다. case by case 이며, 이는 아키텍처 구성에 따라 달라진다.



위와 같은 아키텍처일 경우, 프로그램이 해당 Server 의 Local 캐시와 Remote 캐시를 사용하기 위한 방법은 다음과 같다.

Remote Serve 의 cache.ccf 를 다음과 같이 설정한다.

<server side>

```
registry.host=127.0.0.1
registry.port=1101
remote.cache.service.port=1101
remote.cluster.LocalClusterConsistency=true
remote.cluster.AllowClusterGet=true

jcs.default=
jcs.default.cacheattributes=org.apache.jcs.engine.ComGlueiteCacheAttributes
jcs.default.cacheattributes.MaxObjects=1000000
jcs.default.cacheattributes.MemoryCacheName=org.apache.jcs.engine.memory.lru.LRUMemoryCache
jcs.default.elementattributes=org.apache.jcs.engine.ElementAttributes
jcs.default.elementattributes.IsEternal=false
jcs.default.elementattributes.MaxLifeSeconds=-1
jcs.default.elementattributes.IdleTime=1
jcs.default.elementattributes.IsSpool=true
jcs.default.elementattributes.IsRemote=false
jcs.default.elementattributes.IsLateral=true
```

Application Server 에는 Local 캐시설정 외에, 사용하고자 하는 Remote 캐시 정보를 설정한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

<client side>

```
jcs.default=
jcs.default.cacheattributes=org.apache.jcs.engine.ComGlueiteCacheAttributes
jcs.default.cacheattributes.MaxObjects=1000000
jcs.default.cacheattributes.MemoryCacheName=org.apache.jcs.engine.memory.lru.LRUMemoryCache
jcs.default.cacheattributes.UseMemoryShrinker=true
jcs.default.cacheattributes.MaxMemoryIdleTimeSeconds=3600
jcs.default.cacheattributes.ShrinkerIntervalSeconds=60
jcs.default.elementattributes=org.apache.jcs.engine.ElementAttributes
jcs.default.elementattributes.IsEternal=false
jcs.default.elementattributes.MaxLifeSeconds=700
jcs.default.elementattributes.IdleTime=1800
jcs.default.elementattributes.IsSpool=true
jcs.default.elementattributes.IsRemote=false
jcs.default.elementattributes.IsLateral=true

jcs.region.testCache=RC
jcs.region.testCache.cacheattributes=org.apache.jcs.engine.ComGlueiteCacheAttributes
jcs.region.testCache.cacheattributes.MaxObjects=1000
jcs.region.testCache.cacheattributes.MemoryCacheName=org.apache.jcs.engine.memory.lru.LRUMemoryCache
jcs.region.testCache.elementattributes=org.apache.jcs.engine.ElementAttributes
jcs.region.testCache.elementattributes.IsEternal=false
jcs.region.testCache.elementattributes.MaxLifeSeconds=6000
jcs.region.testCache.elementattributes.IdleTime=1800
jcs.region.testCache.elementattributes.IsSpool=true
jcs.region.testCache.elementattributes.IsRemote=true
jcs.region.testCache.elementattributes.IsLateral=true

jcs.auxiliary.RC=org.apache.jcs.auxiliary.remote.RemoteCacheFactory
jcs.auxiliary.RC.attributes=org.apache.jcs.auxiliary.remote.RemoteCacheAttributes
jcs.auxiliary.RC.attributes.FailoverServers=203.238.195.167:1101
jcs.auxiliary.RC.attributes.RemoveUponRemotePut=true
jcs.auxiliary.RC.attributes.GetOnly=false
jcs.auxiliary.RC.attributes.Receive=true
```

예 제 : JCSRemoteServer-src.zip, JCSTestServer-src.zip

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

Tomcat
정보: Deploying web application archive JcsServer.war
2011-03-31 17:33:15,357 INFO - [main]org.apache.jcs.auxiliary.remote.server.RemoteCacheServerFactor
y!!ConfigFileName = [/cache.ccf]
2011-03-31 17:33:15,357 INFO - [main]org.apache.jcs.auxiliary.remote.server.RemoteCacheServerFactor
y!!Creating server with these attributes
  remoteHost = null
  remotePort = 0
  cacheName = null
  removeUponRemotePut = true
  getOnly = false
  allowClusterGet = true
  localClusterConsistency = true
  getConfigFileName = /cache.ccf
2011-03-31 17:33:15,420 INFO - [main]org.apache.jcs.engine.control.CompositeCacheManager!!Instance
is null, creating with provided config
2011-03-31 17:33:15,420 INFO - [main]org.apache.jcs.engine.control.CompositeCacheManager!!Creating
cache manager from config file: /cache.ccf
2011-03-31 17:33:15,436 INFO - [main]org.apache.jcs.utils.threadpool.ThreadPoolManager!!thread_pool
.default PoolConfiguration = useBoundary = [true] boundarySize = [2000] maximumPoolSize = [150] mini
mumPoolSize = [4] keepAliveTime = [300000] whenBlockedPolicy = [RUN] startUpSize = [4]
2011-03-31 17:33:15,436 INFO - [main]org.apache.jcs.engine.control.CompositeCacheConfigurator!!Sett
ing default auxiliaries to
2011-03-31 17:33:15,436 INFO - [main]org.apache.jcs.engine.control.CompositeCacheConfigurator!!sett
ing defaultCompositeCacheAttributes to [ useLateral = true, useRemote = true, useDisk = true, maxObj
s = 1000000, maxSpoolPerRun = -1, diskUsagePattern = 0 ]
2011-03-31 17:33:15,436 INFO - [main]org.apache.jcs.engine.control.CompositeCacheConfigurator!!sett
ing defaultElementAttributes to [ IS_LATERAL = true, IS_SPOOL = true, IS_REMOTE = false, IS_ETERNAL
= false, MaxLifeSeconds = -1, IdleTime = 1, CreateTime = 1301560395436, LastAccessTime = 13015603954
36, getTimeoutLiveSeconds() = -1, createTime = 1301560395436 ]
2011-03-31 17:33:15,436 INFO - [main]org.apache.jcs.engine.control.CompositeCacheConfigurator!!Pars
ed regions []
2011-03-31 17:33:15,436 INFO - [main]org.apache.jcs.engine.control.CompositeCacheConfigurator!!Finis
hed configuration in 0 ms.
2011-03-31 17:33:15,451 INFO - [main]org.apache.jcs.auxiliary.remote.server.RemoteCacheServerFactor
y!!Binding server to 203.238.195.167:1101 with the name org.apache.jcs.auxiliary.remote.behavior.IRe
moteCacheService

```

[그림 46] Remote Cache Server

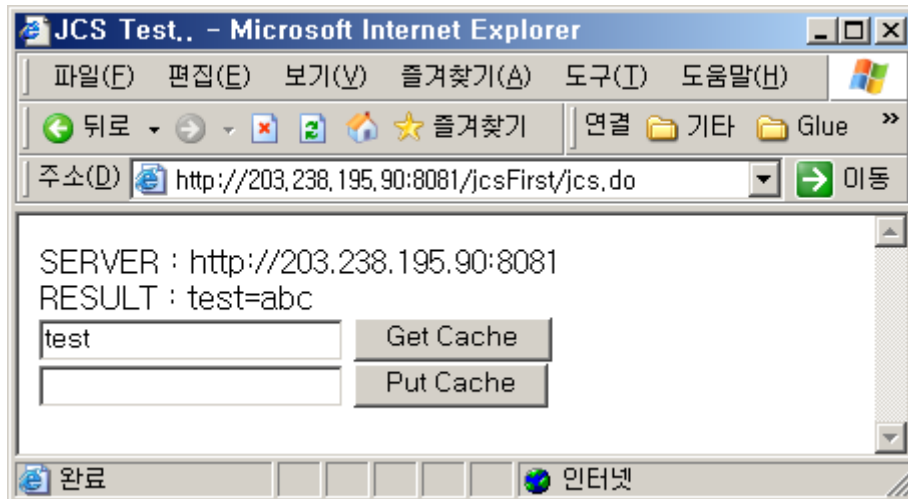
```

Tomcat
정보: Server startup in 17140 ms
2011-03-31 17:44:16,967 INFO - [RMI TCP Connection(6)-203.238.195.90]org.apache.jcs.engine.control.
CompositeCacheConfigurator!!No special ElementAttribute class defined for key [jcs.region.testCache.
elementattributes], using default class.
2011-03-31 17:44:16,982 INFO - [RMI TCP Connection(6)-203.238.195.90]org.apache.jcs.engine.memory.L
ru.LRUMemoryCache!!initialized LRUMemoryCache for testCache
2011-03-31 17:44:16,982 INFO - [RMI TCP Connection(6)-203.238.195.90]org.apache.jcs.engine.control.
CompositeCache!!Constructed cache with name [testCache] and cache attributes [ useLateral = true, us
eRemote = true, useDisk = true, maxObjs = 1000000, maxSpoolPerRun = -1, diskUsagePattern = 0 ]
2011-03-31 17:44:16,998 INFO - [RMI TCP Connection(6)-203.238.195.90]org.apache.jcs.auxiliary.remot
e.server.RemoteCacheServer!!adding vm listener under new id = [1], listenerAddress [203.238.195.90]
2011-03-31 17:44:16,998 INFO - [RMI TCP Connection(6)-203.238.195.90]org.apache.jcs.auxiliary.remot
e.server.RemoteCacheServer!!Region testCache's listener size = 1
2011-03-31 17:45:04,248 INFO - [RMI TCP Connection(8)-203.238.195.90]org.apache.jcs.auxiliary.remot
e.server.RemoteCacheServer!!adding vm listener under new id = [2], listenerAddress [203.238.195.90]
2011-03-31 17:45:04,248 INFO - [RMI TCP Connection(8)-203.238.195.90]org.apache.jcs.auxiliary.remot
e.server.RemoteCacheServer!!Region testCache's listener size = 2

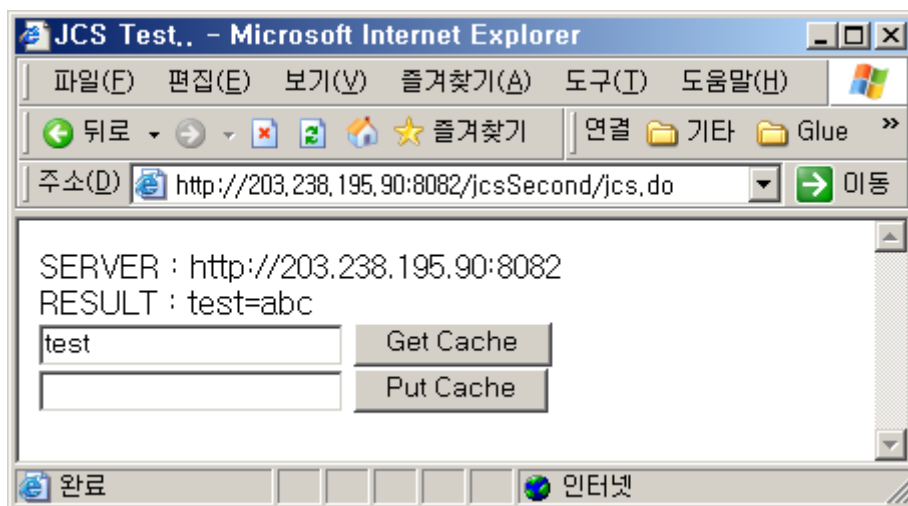
```

[그림 47] Remote Cache Server

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 48] 1 번 서버에서 Put Cache



[그림 49] 2 번 서버에서 Get Cache

7.6 EhCache 개발 가이드

이번 장에서는 Cache Open Source 중 EhCache 을 Glue F/W 에 적용하여 개발하는 과정을 알아보기로 한다.

7.6.1 EhCache

Amazon 과 같이 수 많은 제품을 서비스하는 Site 를 운영한다고 가정해 보자. 전 세계로부터 제품 정보를 조회하는 Service 만 하더라도 DB 서버에는 큰 부담이 될 수 있으며 이를 해결하기 위해 Cache 기능을 생각할 수 있다.

Cache 의 대상은 계절(여름-에어컨, 겨울-장갑) 또는 특수한 날(발렌타인 데이- 초콜릿)에 따라 자주 조회되는 제품이 변경될 수 있기 때문에 자동으로 Cache 에서 Object 가 등록되거나 제거하는 방법 또한 제공이 되어야 한다.

Clustering 된 다수의 WAS 를 지원하기 위한 별도의 Cache Server 또한 필요하다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

이러한 요구사항 외에도 EhCache 는 다양한 기능을 제공하여 서버 퍼포먼스를 높일 수 있는 좋은 Tool 이다.

7.6.2 Spring 에서의 EhCache 활용

EhCache 는 Spring 과 연동 시 개발자가 만든 Business Object(일반적으로 DAO 객체) 에서 Annotation 을 통해 사용한다.

```
@Repository
public class EmpDao
{
    @Cacheable(cacheName="empCache")
    public Employee findEmployee(String empno)
    {
        // 실제 Emp 를 조회하는 로직
        Employee emp = . . . . .

        return emp;
    }
    . . . . .
}
```

[Spring3 + EhCache + Annotation]으로 검색하면 자세한 사용법 등을 얻을 수 있다.

7.6.3 Glue Framework 에서의 EhCache 활용

하지만 Glue Framework 에서는 Service Object 와 DAO 객체를 개발자가 개발하지 않고 Activity Diagram 을 이용하여 개발하기 때문에 이를 직접 Handling 할 수 없다.

따라서 Glue Framework 에서는 EhCache 에 직접 접근하여 Data Handling 하여야 하며 아래와 같이 제공할 예정이다.

Cache 기능을 가진 Reusable Activity Class 의 제작

Custom Activity Class 에서 개발자가 직접 Cache 에 등록

Hibernate 를 통한 EhCache 연동

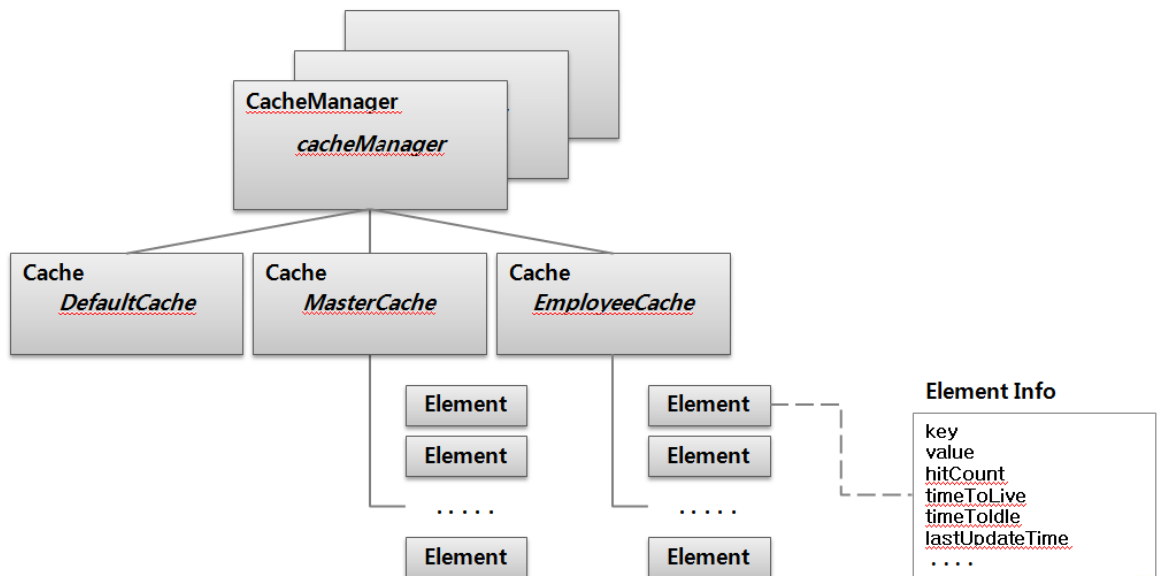
7.6.4 Business 적용

업무 분야에서 Caching 대상은 설계자 및 개발자의 몫으로 어떤 Business Object 를 어떤 정책을 가지고 제공할 것인지 생각해 볼 필요가 있다.

7.6.5 EhCache 의 구조

EhCache 의 구분은 managerName > cacheName > cacheKey 와 같이 3 단계로 구분된다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 50] Ehcache

아래는 위 그림에 대한 CacheManager 설정 예이다.

```

<ehcache:config cache-manager="cacheManager">
  <ehcache:evict-expired-elements interval="300" />
</ehcache:config>

<bean id="cacheManager"
class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean">
  <property name="configLocation" value="/WEB-INF/cache/employeeCache.xml" />
</bean>

```

[그림 51] EhCache

Cache Object 에 대한 설정 예

```

<?xml version="1.0" encoding="UTF-8"?>
<ehcache . . . updateCheck="false">

  <cache name="MasterCache"
    maxElementsInMemory="100"
    eternal="false"
    overflowToDisk="false"
    timeToLiveSeconds="30"
  />

  <cache name="EmployeeCache"
    maxEntriesLocalHeap="1000"
    eternal="false"
    timeToIdleSeconds="3000"
    timeToLiveSeconds="1800"
    overflowToDisk="false"
  />
</ehcache>

```

[그림 52] EhCache

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

EhCache 에는 JCS 와는 다르게 CacheManager 라는 layer 가 하나 더 존재하며 cacheManagerName 으로 각 Manager 를 구분할 수 있다

JCS 의 Region 에 해당하는 Cache 는 CacheName 으로 구분할 수 있다.

Cache 에서 Cachekey 로 해당 Caching 한 Object 를 가져올 수 있으며 이 Object 는 Element 라는 객체 내 존재한다.

EhCache 는 Spring 과 연동하여 Cache 기능을 수행하기

EhCache Manager 를 사용하기 위해서는 두 가지 선행 작업이 필요하다. 첫 번째로

applicationContext.xml 설정 파일에 'cacheManager'란 아이디를 갖는 Bean 이 등록되어 있어야 한다.

7.6.6 EhCache 설정

7.6.6.1 EhCacheManager 정의

```
<bean id="cacheManager"
class="com.poscoict.glueframework.cache.ecache.GlueEhCacheManager"/>
```

7.6.6.2 EhCacheManagerFactoryBean 정의

applicationContext.xml 또는 dispatcher-servlet.xml 에 아래와 같이 추가한다.

```
<ehcache:annotation-driven/>
<ehcache:config cache-manager="ehCache">
<!-- Currently set on 1 day (1440 minutes) -->
  <ehcache:evict-expired-elements interval="1440" />
</ehcache:config>
<bean id="ehCache"
class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean">
  <property name="configLocation" value="/WEB-INF/cache/employeeCache.xml"
/>
</bean>
```

7.6.6.3 Cache 상세 설정

실제 Cache 정의는 configLocation 에서 지정한 파일에 작성한다

/WEB-INF/cache/employeeCache.xml 에 추가 내용(employeeCache.xml 은 예제임)

```
<cache name="emp"
maxElementsInMemory="100"
eternal="false"
overflowToDisk="false"
timeToLiveSeconds="600" />
```

XML 내용은 Cache 의 성격에 맞게 작성하며 자세한 내용은

<http://ehcache.org/documentation/user-guide/configuration> 를 참고하여 작성한다.

GlueFramework	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

8. Logging

8.1 정의 및 역할

잘 작성된 애플리케이션은 에러가 발생하여 디버깅을 해야 하는 시점에 에러가 발생한 곳이 어디이며, 어디까지 실행이 되다가 어떤 원인 때문에 발생했는지에 대한 세부정보를 제공해준다. 이와 같은 처리를 지원해 주는 것이 Logging 이다.

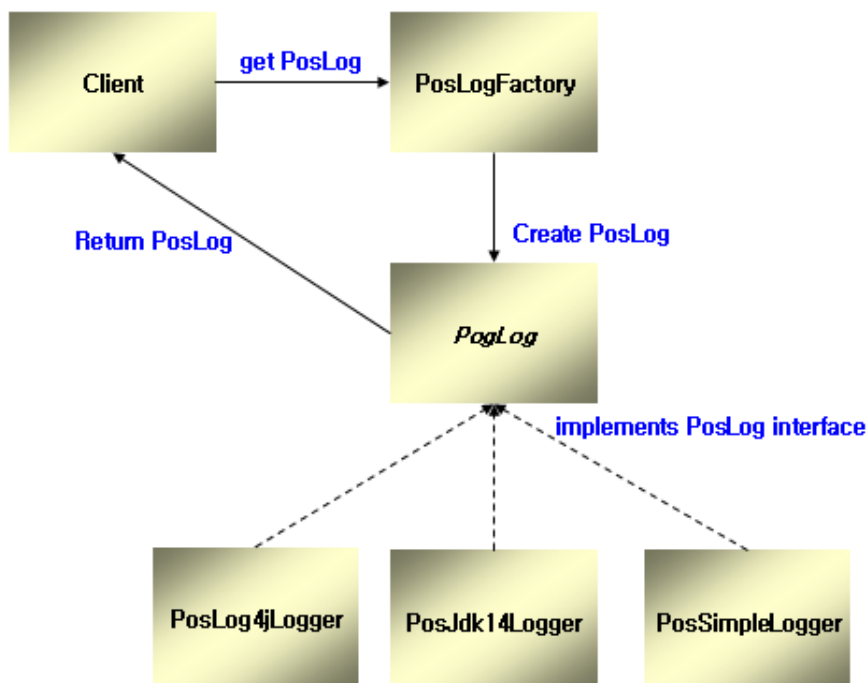
애플리케이션을 개발할 때 체계적이고 상세한 Logging 메시지는 애플리케이션을 개발하는 도중뿐만 아니라 운영하는 시점에도 상당한 위력을 발휘한다.

8.2 작동 방식

Glue Framework 의 Logging Component 는 Log4J 와 JDK1.4 Logger 의 두 가지 기능을 사용할 수 있도록 지원해 주는 GlueLog 라는 인터페이스를 제공해준다.

현재 동작하는 애플리케이션의 클래스패스에 Log4J 라이브러리가 존재하면 log4j Logger 가 로딩되어 동작하고 그렇지 않고 JDK1.4 Logger(java.util.logging.Logger)가 존재하면 JDK1.4 Logger 가 작동한다. 만약 JDK1.4 미만 버전을 사용하는 환경이라면 GlueSimpleLogger 라는 클래스가 작동한다. Glue Framework 의 표준환경은 JDK1.4 이상에서 Log4J 라이브러리를 사용하는 것이다.

Log4J 는 log4j.xml 이란 환경설정 파일을 사용하고, JDK1.4 Logger 는 logging.properties 환경설정 파일을 사용한다. 사용하고자 하는 파일을 클래스패스 Root 에 위치시키면 된다.



[그림 53] Logger

8.3 개발 가이드

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

8.3.1 Log4J 를 사용하는 경우

설정파일(log4j.xml)을 다음과 같이 작성한다.

```
<?xml version="1.0" encoding="euc-kr" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j='http://jakarta.apache.org/log4j/'>
  <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern"
        value = "%d %-5p - [%t]%C| |%m%n"/>
    </layout>
  </appender>
  <appender name="FILEAPPENDER"
    class="org.apache.log4j.RollingFileAppender">
    <param name="append" value="true"/>
    <param name="file" value="D:/Temp/glue_debug_log.log"/>
    <param name="maxFileSize" value="5MB"/>
    <param name="maxBackupIndex" value="100"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern"
        value = "%d %-5p - [%t]%C| |%m%n"/>
    </layout>
  </appender>
  <logger name="org.apache">
    <level value="info"/>
  </logger>
  <logger name="org.springframework">
    <level value="info"/>
  </logger>
  <logger name="com.poscoict.glueframework">
    <level value="debug"/>
  </logger>
  <root>
    <level value="debug"/>
    <appender-ref ref="FILEAPPENDER"/>
  </root>
</log4j:configuration>
```

log4j.xml 에는 logger, appender 들로 구성되어 있다. CONSOLE, FILEAPPENDER appender 등을 정의할 수 있다. CONSOLE 를 통해 표준출력장치로 로깅 (보통 콘솔에 출력 됨)하고, FILEAPPENDER 을 통해 file 속성에 정의되는 파일에 로그가 기록된다. 각 appender 의 ConversionPattern 을 통해 로그가 출력되는 포맷을 정의할 수 있다. org.apache logger 를 통해 Apache 관련 라이브러리의 로그 레벨을 정의한다. org.springframework logger 을 통해 Spring Framework 관련 라이브러리의 로그 레벨을 정의한다. com.poscoict.glueframework logger 를 통해 Glue Framework 관련 라이브러리의 로그 레벨을 정의한다. root 항목을 통해 User Application 포함한 나머지 라이브러리에 대한 로그 레벨을 정의한다.

8.3.2 JDK1.4 Logger 를 사용하는 경우

설정파일은 %JDKHome%\jre\lib\logging.properties 에 위치한다.

Log4J 와 같은 특정 패키지별로 로그 레벨 설정을 지원하지 않는다.

설정파일(logging.properties)을 다음과 같이 작성한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
#####
#      Default Logging Configuration File
# You can use a different file by specifying a filename
# with the java.util.logging.config.file system property.
# For example java -Djava.util.logging.config.file=myfile
#####
#####
#      Global properties
#####
# "handlers" specifies a comma separated list of log Handler
# classes.  These handlers will be installed during VM startup.
# Note that these classes must be on the system classpath.
# By default we only configure a ConsoleHandler, which will only
# show messages at the INFO and above levels.
handlers= java.util.logging.ConsoleHandler

# To also add the FileHandler, use the following line instead.
#handlers= java.util.logging.FileHandler, java.util.logging.ConsoleHandler

# Default global logging level.
# This specifies which kinds of events are logged across
# all loggers.  For any given facility this global level
# can be overridden by a facility specific level
# Note that the ConsoleHandler also has a separate level
# setting to limit messages printed to the console.
.level= INFO

#####
# Handler specific properties.
# Describes specific configuration info for Handlers.
#####

# default file output is in user's home directory.
java.util.logging.FileHandler.pattern = %h/java%u.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.formatter = java.util.logging.XMLFormatter

# Limit the message that are printed on the console to INFO and above.
java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter

#####
# Facility specific properties.
# Provides extra control for each logger.
#####

# For example, set the com.xyz.foo logger to only log SEVERE
# messages:
com.xyz.foo.level = SEVERE
```

8.3.3 Logging Component 사용 예제

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

런타임에 사용되는 Logger 가 결정되기 때문에 구현되는 소스 코드 스타일은 Logger 에 상관 없이 항상 동일하다.

```
import com.poscoict.glueframework.GlueException;
import com.poscoict.glueframework.util.log.GlueLog;
import com.poscoict.glueframework.util.log.GlueLogFactory;

public class LoggingClient
{
    // 각 클래스별로 GlueLogger 를 등록함
    protected GlueLog logger =
        GlueLogFactory.getLogger(LoggingClient.class);
    public LoggingClient()
    {
    }
    public void performTask()
    {
        try
        {
            // 현재 로그 레벨이 디버그 이상이면
            // (로그레벨, debug < info < warn < error < fatal)
            if (logger.isDebugEnabled())
            {
                // debug logging
                logger.logDebug("debug logging test");
            }
            // 로깅할 메시지가 info 레벨 이상이 되는 경우는
            // 현재 로그 레벨을 체크하지 않아도 상관없다.
            logger.logInfo("info logging test");
        }
        catch (GlueException ex)
        {
            // 에러 메시지와 Exception Stacktrace 를 로깅한다.
            logger.logError("Fail to perform task: " + ex.getMessage(), ex);
        }
    }
}
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

9. Exception 처리

9.1 정의 및 역할

예외(Exception)란? 의도한 대로 프로그램이 진행되지 않고 다른 상황으로 흐름이 진행되는 것을 말한다. 이런 예외가 발생했을 때 적절한 조치를 취해 줄 수 있는 절차를 구현하도록 문법을 정의한 것을 예외 처리라 한다.

자바가 다른 언어와 크게 다른 것이 이 예외를 처리하는 과정이다. 자바에서는 Checked Exception 과 Unchecked Exception 두 가지 방법을 지원함으로써 예외 처리를 컴파일 시점과 런타임 시점에서 처리하는 것이 가능하도록 한다.

Exception 의 상황에 따라 적절히 사용되어야 함에도 불구하고 지금까지 자바 진영에서는 Checked Exception 의 장점만을 너무 강조함으로써 Checked Exception 을 무분별하게 사용하고 있다. 무분별한 Checked Exception 의 사용은 다음과 같은 단점을 가져 온다

- 소스 코드의 복잡도 증가
- 소스 코드의 가독성(Readability) 저하, 유지보수 저하
- 비즈니스적인 Exception 을 제외하고 개발자가 처리할 수 있는 것은 상당히 제한적

9.2 Checked Exception vs Unchecked Exception

자바의 Checked Exception 은 java.lang.Exception 을 상속한 Exception 으로 컴파일 시점에 이 Exception 을 try~catch 구문으로 처리하지 않을 경우 컴파일 에러가 발생한다. 그러므로, 컴파일 시점에 메소드에서 발생하는 Exception 처리가 제대로 구현되어 있는지를 확인하는 것이 가능하다. 예를 들어 java.io.IOException 이 대표적인 Checked Exception 이다.

Unchecked Exception 은 java.lang.RuntimeException 을 상속한 Exception 으로 컴파일 시점에 try~catch 구문으로 처리하지 않아도 컴파일 에러가 발생하지 않는다. 단, 런타임 시점에 에러가 발생할 경우 Unchecked Exception 이 발생하게 되는 것이다. 예를 들어, java.lang.NullPointerException 이 대표적인 Unchecked Exception 이다.

Source Code Example	
Checked Exception	Unchecked Exception
<pre> public Task getTask() throws Exception; public void performTask() { try { // ... pre processing Task task = getTask(); task.start(); // ... post processing } catch (Exception e) { // handle exception } } </pre>	<pre> public Task getTask() throws RuntimeException; public void performTask() { Task task = getTask(); task.start(); } </pre>

[그림 54] Checked Exception vs Unchecked Exception

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

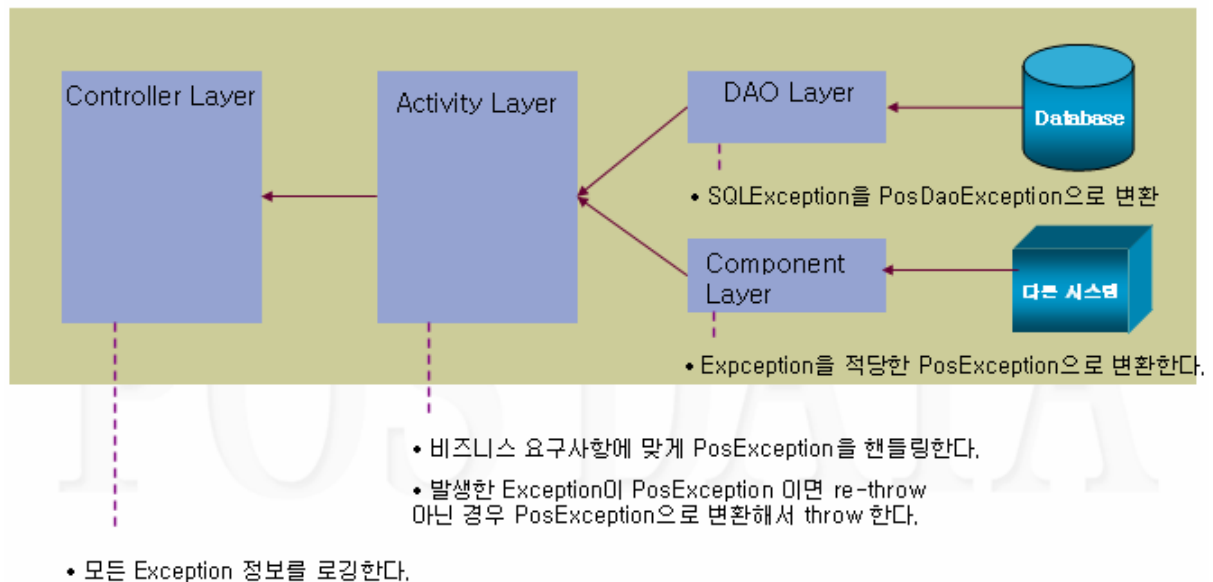
9.3 Glue Framework에서의 Exception 처리

9.3.1 Exception 처리 전략

Glue Framework에서는 다음과 같은 경우를 제외하고는 Unchecked Exception (com.poscoict.glueframework.GlueException)을 사용한다.

- 회복 가능한 에러가 발생한 경우
- 모든 호출자들이 발생 되는 Exception을 꼭 처리해야 하는 경우 → 비즈니스적인 Exception

9.3.2 각 Layer 별 Exception 처리



[그림 55] 각 Layer 별 Exception 처리

Activity Layer는 바로 GlueActivity를 상속하여 구현한 각 비즈니스 도메인의 비즈니스 클래스이다. 비즈니스 클래스에서는 별다른 Exception 처리가 필요하지 않은 경우에는 try ~ catch 구문이 필요 없다. 다만 비즈니스적으로 다른 로직을 수행하고자 한다면 GlueException을 상속받아 신규 Exception 클래스를 정의한 후에 처리를 해 주면 된다. 물론 Glue Framework에서 제공하는 [Exception 오브젝트](#)만으로도 Exception 처리가 가능하다면 신규 Exception 클래스가 필요없다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

import com.poscoict.glueframework.biz.activity.GlueActivity;
import com.poscoict.glueframework.biz.constants.GlueBizControlConstants;
import com.poscoict.glueframework.context.GlueContext;

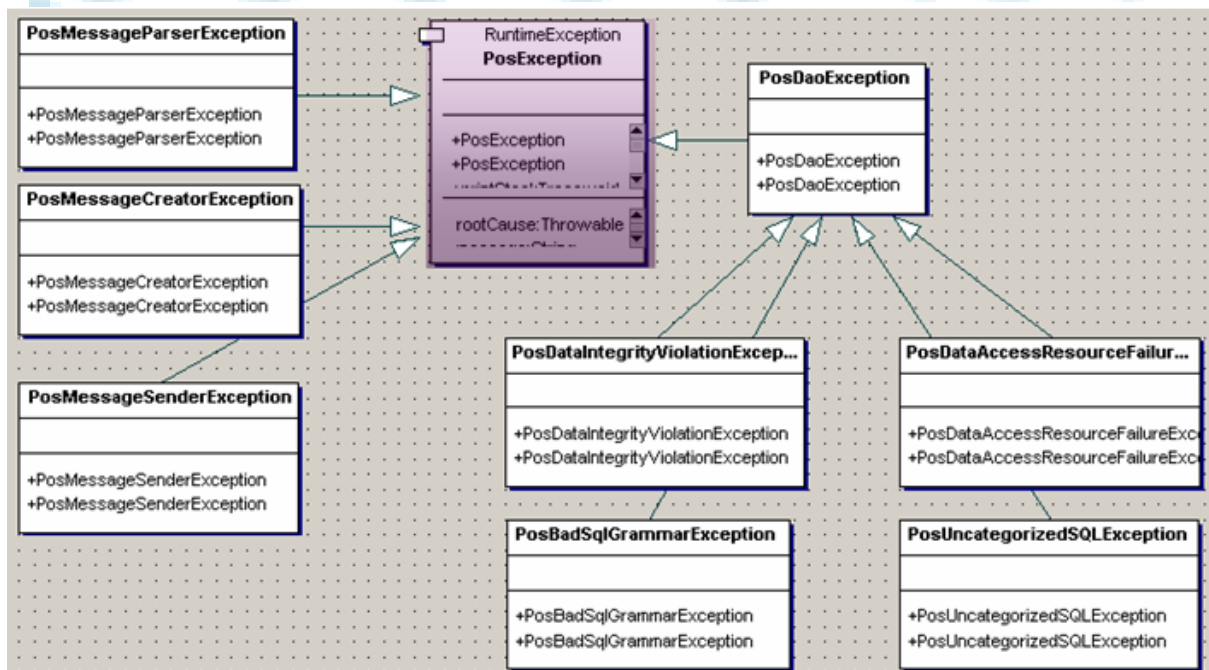
public class MyActivity extends GlueActivity
{
    public String runActivity(GlueContext ctx)
    {
        try
        {
            // execute Biz Logic
        }
        catch (GlueUserDefinedException ex)
        {
            // handle exception
        }
        return GlueBizControlConstants.SUCCESS;
    }
}

```

9.3.3 GlueException 계층도

GlueException 은 java.lang.RuntimeException 을 확장한 Unchecked Exception 으로서 Glue Framework 에서 사용하는 최상위 Exception 이다. 컴포넌트 관련 Exception 과 DAO 관련 Exception 은 GlueException 을 확장하여 구현되었다.

별다른 요건이 없는 경우(모든 호출자들이 발생하는 Exception 을 항상 처리해야 하는 경우)는 항상 GlueException 을 확장하여 구현한다.



[그림 56] Exception

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

9.3.4 GlueDaoException 처리 방법

DAO 컴포넌트에서 발생하는 Exception 의 유형은 총 4 종류이다. 유형별 발생하는 케이스는 다음 테이블을 참조하기 바란다.

DAO Exception 유형	발생 경위
GlueDataIntegrityViolationException	Database 의 레코드에 insert, update 수행 중에 integrity constraint 에 위배되는 경우
GlueBadSqlGrammarException	SQL 문법이 틀린 경우 발생
GlueDataAccessResourceFailureException	Resource(Database)접근에 실패한 경우
GlueUncategorizedSQLException	SQLException 에 관련된 나머지 에러들

[표 14] GlueDaoException

GlueDaoException 을 확장하여 구현된 상기 4 개의 오브젝트는 에러가 발생할 당시에 수행 된 쿼리를 포함한다. GlueBadSqlGrammarException 와 GlueUncategorizedSQLException 은 SQL 에러코드까지 포함한다.

이미 언급한 것처럼 Exception 이 발생하여도 별다른 처리가 필요하지 않다면 try~catch 문이 필요 없다.

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

10. Web 화면 개발

Glue에서는 Web 개발에 기본적으로 Spring MVC를 사용하고 있다.

10.1 DispatcherServlet 정의

Web.xml에 정의한다.

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/mvc-servlet.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

10.2 HandlerMapping 설정

위 Web.xml의 dispatcher servlet에 정의된 xxx-servlet.xml에 handler mapping 정보를 정의한다.

10.2.1 BeanNameUrlHandlerMapping

요청 URI와 동일한 이름을 가진 Controller 빈을 매핑한다.

```
<bean id="beanNameUrlMapping"
  class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />
<bean name="/hello.do" class="com.glueframework...GlueSimpleController" />
<bean name="/login.do" class="com.glueframework...GlueLoginController" />
. . .
```

10.2.2 SimpleUrlHandlerMapping

Ant 스타일의 경로 매핑 방식을 사용하여 URI와 Controller 빈을 매핑한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
<bean id="beanNameUrlMapping"
      class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <value>
      /content/*.do=contentController
      /**/help.do=helpController
    </value>
  </property>
</bean>
<bean name="contentController"
      class="com.glueframework...GlueSimpleController " />
<bean name="helpController"
      class="com.glueframework..HelpController" />
```

10.3 ViewResolver 설정

10.3.1 ViewResolver 역할

Controller 는 요청을 처리 한 뒤 ModelAndView 객체를 넘겨준다. 이 때 이 객체에 view 의 이름을 같이 넘겨 주는데 이 이름으로부터 실제 view 를 찾아 주는 역할을 하는 것이 View Resolver 이다.

```
<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix" value="/view/" />
  <property name="suffix" value=".jsp" />
</bean>
```

*prefix - Controller 가 리턴한 뷰 이름 앞에 붙을 접두어

*suffix - Controller 가 리턴한 뷰 이름 뒤에 붙을 확장자

10.3.2 ViewResolver 종류

ViewResolver	설명
AbstractCachingViewResolver	View 들을 caching 하는 기능 제공
XmlViewResolver	ViewResolver 의 구현체로 XML 파일 사용(/WEB-INF/views.xml 을 기본 설정파일로 사용)
ResourceBundleViewResolver	ViewResolver 의 구현체로 리소스 파일 사용(views.properties 를 기본 리소스 파일로 사용)
UrlBasedViewResolver	ViewResolver 의 구현체로 특별한 맵핑 정보 없이 의미상 view 이름을 URL 로 사용(View 이름과 실제 view 자원과의 이름이 같을 때 사용)
InternalResourceViewResolver	UrlBasedViewResolver 를 상속 받았으며 InternalResourceView (Servlet, JSP)를 사용
VelocityViewResolver / FreeMarkerViewResolver	UrlBasedViewResolver 를 상속 받았으며 VelocityView 와 FreeMarkerView 를 사용

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

[표 15] ViewResolver

10.4 Controller

10.4.1 GlueAbstractController 사용

Glue Service 를 실행하기 위해서 필요한 기본적인 로직(GlueContext 생성등)이 구현된 Abstract Class

GlueAbstractController 상속 받을 경우 구현해야 할 메서드는 아래와 같다.

*preDoAction - 서비스 실행 전 실행되는 메서드

*afterDoAction - 서비스 실행 후 실행되는 메서드

*setModelAndView - return 할 ModelAndView 생성

```
public abstract void preDoAction(GlueWebContext ctx);
public abstract void afterDoAction(GlueWebContext ctx);
public abstract ModelAndView setModelAndView(GlueWebContext ctx);
```

10.4.2 GlueSimpleController

GlueAbstractController 을 상속받아서 구현된 Controller.

해당 서비스 실행 전후 특별히 추가해야 할 로직이 없다면 GlueSimpleController 를 사용하면 된다.

해당 서비스 실행 후 view page 가 설정되어 있지 않으면 Url 을 기초로 view page 를 설정한다(예: url 이 dept.do 이면 view page 는 dept 로 설정)

10.4.3 Exception 발생 시 View Page(GlueAbstractController)

Controller 의 property 로 errorPage 가 설정 되어 있으면 errorPage 를 View Page 로 설정한다.

errorPage 가 설정 되어 있지 않으면 viewPage 값이 View Page 로 설정된다.

```
<bean name="controller"
      class="com.posdata.glue.web.control.GlueSimpleController" />
  <property name="viewPage" value="dept" />
  <property name="errorPage" value="error" />
</bean>
```

10.5 Interceptor

Interceptor 를 사용하면 Controller 가 요청을 처리하기 전 후에 기능을 수행하도록 설정이 가능하다.

10.5.1 Interceptor 구현

HandlerInterceptorAdapter 을 상속받아서 구현한다. 다음 3 가지 메소드를 구현할수 있다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

- boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
- void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler , ModelAndView modelAndView)
- void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)

```
public boolean preHandle(HttpServletRequest request,
                        HttpServletResponse response,
                        Object handler) throws Exception
{
    log.debug("==== preHandle Interceptor =====");
    if(request.getSession().getAttribute("userId") != null)
    {
        return true;
    } else {
        response.sendRedirect("login.jsp");
        return false;
    }
}
```

요청 컨트롤러 수행전에 수행된다. false 를 return 하면 그 다음 interceptor 또는 컨트롤러 수행하지 않는다.

```
public void postHandle(HttpServletRequest httpServletRequest,
                      HttpServletResponse httpServletResponse,
                      Object obj,
                      ModelAndView modelAndView) throws Exception
{
    /** request 로 넘어온 데이터를 가공할때 쓰는 곳이다 */
    log.debug("==== postHandle Interceptor =====");
}
```

그 전 interceptor 수행 또는 컨트롤러 수행중에 예외 발생시 수행 안함

```
public void afterCompletion(HttpServletRequest httpServletRequest,
                           HttpServletResponse httpServletResponse,
                           Object obj,
                           Exception exception) throws Exception
{
    /** 공통 Exception 처리를 해주는것이 좋다. 에러 메일을 보내는것등 */
    log.debug("==== afterCompletion Interceptor =====");
}
```

뷰를 통해서 클라이언트에게 응답을 전송한 뒤 수행

10.5.2 Interceptor 설정

xxx-servlet.xml 에서 HandlerMapping 클래스에 Interceptors 값을 설정해 줄 수 있다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<bean id="handlerMapping"
class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="interceptors">
    <list>
      <ref bean="commInterceptor"/>
    </list>
  </property>
  <property name="mappings">
    <props>
      .....
    </props>
  </property>
</bean>
<bean id="commInterceptor" class="common.CommInterceptor"></bean>

```

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

11. Non-UI 개발

11.1 Glue Framework 설정

11.1.1 Glue Non-UI 환경파일 구성

TC Layout 파일과 Sending 파일 부분을 제외한 나머지 부분은 개발환경 설정 부분과 동일하다.

구분	위치	환경파일
TC Layout File	WEB-INF\classes\layout	XXX-msg.xml

[표 16] Layout

11.1.2 Glue Framework 설정

11.1.2.1 applicationContext.xml 설정

Message component 관련 설정은 아래와 같으며, applicationContext.xml 파일에 추가한다.

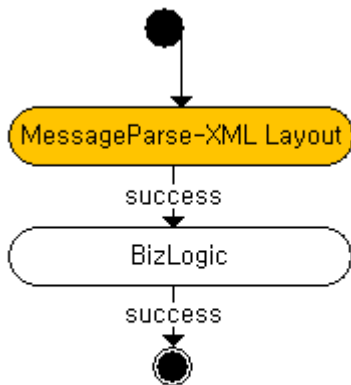
1) Message Layout 설정

```
<!--XML Msg Layout-->
<bean id="layoutInXml" class="
com.glueframework.message.layout.GlueXmlMessageLayout"/>
<!--DB Msg Layout-->
<bean id="layoutInDB" class="
com.glueframework.message.layout.GlueDBMessageLayout "/>
```

11.2 설계서 및 설계 산출물

11.2.1 Activity Diagram

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 57] Activity Diaram

Message Receiver 를 통해 수행되는 Activity Diagram 은 Transition 이 반드시 1 개여야 한다. 즉 수행 흐름은 1 가지만 존재해야 한다. 하지만 중간에 Router 사용은 가능하다.

11.2.2 Service XML 파일

```

<?xml version="1.0" encoding="UTF-8"?>
<service name="MSGFW002-service" initial="MessageParse-XML Layout">
  <activity name="BizLogic" class="sample.activity.BizLogic">
    <transition name="success" value="end" />
  </activity>
  <activity name="MessageParse-XML Layout"
    class="com.glueframework.biz.activity.GlueMessageParse">
    <transition name="success" value="BizLogic" />
    <property name="layout" value="layouInXml" />
  </activity>
</service>

```

11.2.3 Query 파일

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<?xml version="1.0" encoding="EUC-KR"?>
<queryMap desc="EMP Project 쿼리 파일">
  <query id="emp.insert" desc="사원추가" >
    <![CDATA[
insert into emp(empno, ename, sal,deptno)
  values(?, ?, ?,?)
    ]]>
  </query>
  <query id="emp.update" desc="사원정보수정" >
    <![CDATA[
update emp
  set sal=?,ename=?
  where empno=?
    ]]>
  </query>
  <query id="emp.select" desc="사원조회" fetchSize="10">
    <![CDATA[
select EMPNO, ENAME, JOB, MGR, HIREDATE, SAL,DEPTNO
  from emp where DEPTNO=?
    ]]>
  </query>
  <query id="emp.select.dynamic" desc="dynamic where 쿼리 test" fetchSize="10">
    <![CDATA[
select
  EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO
from emp
    ]]>
    <where>
      <set conjunction="and">
        <attribute conjunction="or"><![CDATA[ JOB!=? ]]></attribute>
        <attribute conjunction="end"><![CDATA[ DEPTNO=? ]]></attribute>
      </set>
      <set conjunction="end">
        <attribute conjunction="or"><![CDATA[ EMPNO=? ]]></attribute>
        <attribute conjunction="and"><![CDATA[ EMPNO=? ]]></attribute>
        <attribute conjunction="end"><![CDATA[ EMPNO=? ]]></attribute>
      </set>
    </where>
    <orderby><![CDATA[empno ]]></orderby>
  </query>
</queryMap>

```

[그림 58] Query

단, DB 를 처리할 부분이 있을 경우 필요하다.

11.2.4 Message Layout XML 파일

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
<?xml version="1.0" encoding="EUC-KR"?>
<msgs>
  <msg id="MSGEMP01" name="receiver 테스트"><!--TC id-->
    <attribute type="E" seq="1" id="TRANSACTION_CODE" name="TransactionCode"
      datatype="STRING" length="8" precision="" />
    <attribute type="E" seq="2" id="WORKS_CODE" name="사소구분"
      datatype="STRING" length="1" precision="" />
    <attribute type="E" seq="3" id="OPER_FLAG" name="조업구분" datatype="STRING"
      length="1" precision="" />
    <attribute type="E" seq="4" id="FAC_OP_CD" name="공장공정코드"
      datatype="STRING" length="2" precision="" />
    <attribute type="E" seq="5" id="SNDR_INFORM_EDIT_DATE"
      name="송신측정보편성일시" datatype="STRING" length="14" precision="" />
    <attribute type="E" seq="6" id="SNDR_INFORM_EDIT_PGM_ID"
      name="송신측정보편성ProgramID" datatype="STRING" length="14" precision="" />
    <attribute type="E" seq="7" id="EAI_INTERFACE_ID"
      name="EAI송수신관리InterfaceID" datatype="STRING" length="12" precision="" />
    <attribute type="E" seq="8" id="INTERFACE_DATA_DIR_ACTUAL_TYPE"
      name="InterfaceData지시실적구분" datatype="STRING" length="1" precision="" />
    <attribute type="E" seq="9" id="INTERFACE_DATA_OCR_RES_FLAG"
      name="InterfaceData발생응답구분" datatype="STRING" length="1" precision="" />
    <attribute type="E" seq="10" id="INTERFACE_DATA_SEND_SEQ"
      name="InterfaceData송신순서" datatype="NUMBER" length="5" precision="" />
    <attribute type="E" seq="11" id="INTERFACE_DATA_UPD_TP"
      name="InterfaceData수정구분" datatype="STRING" length="1" precision="" />
    <attribute type="E" seq="12" id="INTERFACE_DATA_T_LEN"
      name="InterfaceData총길이" datatype="NUMBER" length="6" precision="" />
    .....중략.....
    <attribute type="E" seq="21" id="HIREDATE" name="HIREDATE" datatype="DATE"
      length="8" precision="" />
    <attribute type="E" seq="22" id="COMM" name="COMM" datatype="NUMBER"
      length="7" precision="2" />
  </msg>
</msgs>
```

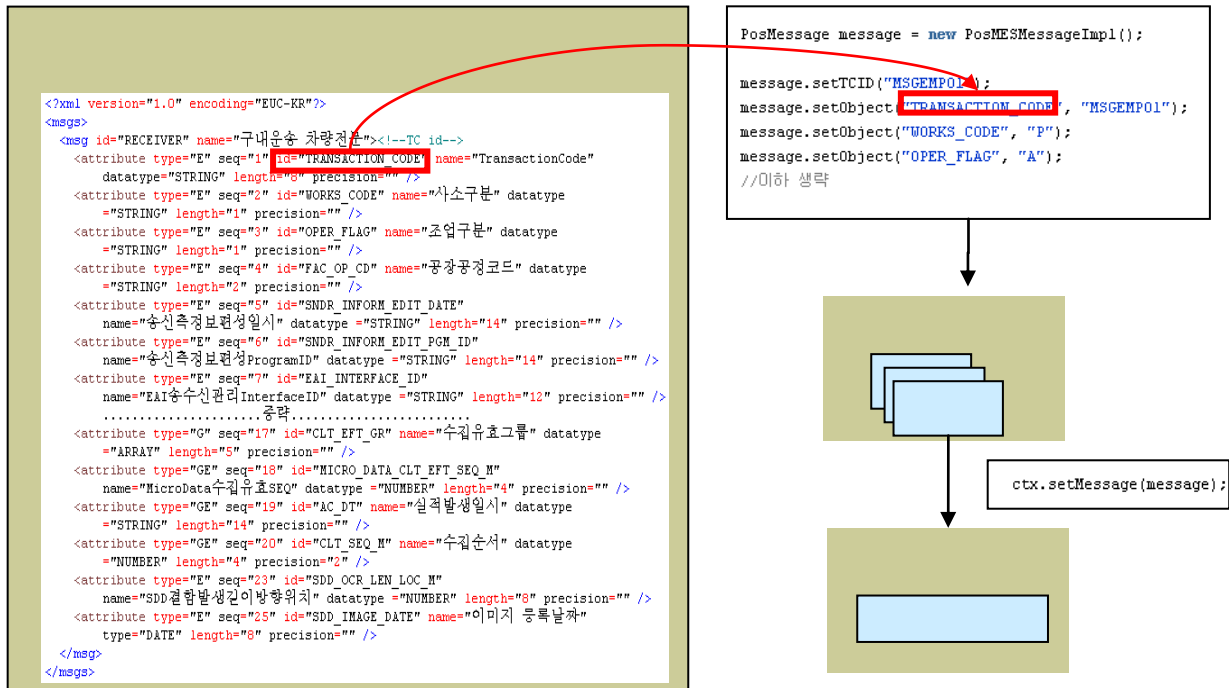
[그림 59] MessageLayout

11.2.5 Message Layout XML 파일과 Message Data Handling

Framework 에서 사용되는 모든 Data 는 PosContext 객체에 저장한다. Message Data 도 PosContext 객체에 저장된다.

11.2.5.1 Message(TC) 송수신

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 60] MessageParse

posco ict

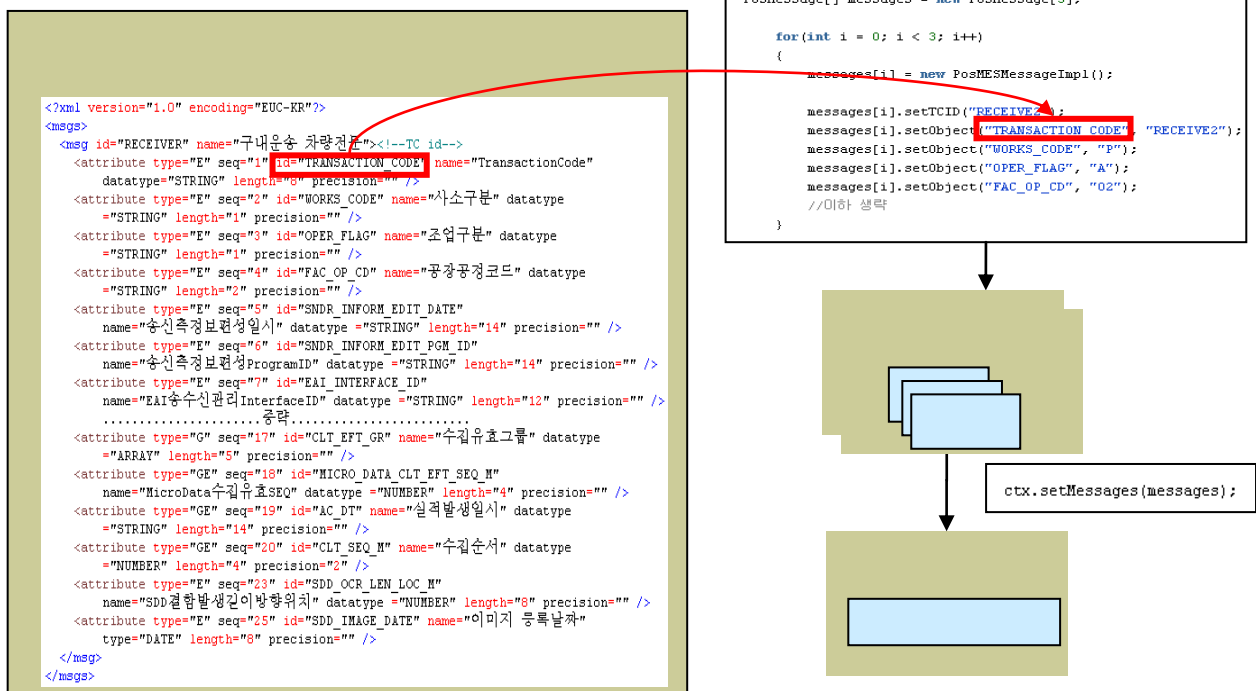
Message Layout 에 정의된 Id 로 PosMessage 객체에 Message Data 를 저장 후 PosContext 에 저장한다. Message Data 를 가져올 때도 동일하게 PosContext 에서 꺼내 PosMessage 객체에서 가져온다.

Message Layout 에 Type 이 Group 으로 설정된 경우 Group Element 는 Group 의 Length 에 정의된 반복 횟수만큼 id+Sequence 로 Key 값을 정의하여 저장한다.

(예 : message.setObject("CLT_LET_SEQ", "20001"))

11.2.5.2 Message(File) 송수신

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 61] MessageParse

poscoict

Message Layout 에 정의된 Id 로 PosMessage 객체에 Array 로 정의하여 Message Data 를 저장 후 PosContext 에 저장한다. Message Data 를 가져올 때도 동일하게 PosContext 에서 꺼내 PosMessage[] 객체에서 가져온다.

File Message 는 동일 Message Layout 에 여러 TC 를 송수신시 정의하여 사용한다.

11.3 개발 범위 및 방법

11.3.1 개발 범위

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Message Receiver 를 통해 Message 를 수신되는 경우 Message ID(TC ID)에 따라 GlueBizController 를 통한 Business Logic 을 수행한다.

GlueBizController 는 Non-UI, UI 에서 동일하게 수행된다.

Message Component 은 Activity 및 Method 단위로 실행 가능하다.

11.3.2 개발 방법

먼저 수행할 Activity Diagram 을 설계하고, 사용될 TCID-service.xml, TCID-msg.xml, XXX-quary.xml 를 만든다. 동일 TC ID 로 생성된 Message Layout 은 Non-UI, UI 모두 동일하게 사용된다.

각 Activity 에 해당하는 Business Logic 을 구현한다.

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

12. Globalization & Multi TimeZone

12.1 Globalization

12.1.1 Glue Framework 설정

12.1.1.1 applicationContext.xml 설정

Spring 에서 제공하는 MessageSource 를 사용하기 위해서 applicationContext.xml 에 Bean 을 등록한다.

```
<bean id="messageSource"
      class="org.springframework.context.support.ResourceBundleMessageSource">
  <property name="basenames">
    <list>
      <value>message</value>
    </list>
  </property>
</bean>
```

12.1.1.2 properties 파일 작성

Locale 별 Property 파일의 명명 규칙은 다음과 같다. 그중 language, country 가 포함된 2 번째 규칙을 기본으로 하나, Locale 정보에 따라 그외도 가능하다.

- <Resource 명> "_" language "_" country "_" variant
- <Resource 명> "_" language "_" country
- <Resource 명> "_" language
- <Resource 명>

따라서 다음과 같은 형태의 이름들을 가질 수 있다.

- message_ko_KR_IBMeucKR.properties
- message_ko_KR.properties
- message_ko.properties
- message_en_US.properties
- message_en.properties
- message.properties

우리나라와 같이 2byte 문자를 사용하는 경우는 MessageSource 에 변경사항이 발생할 때마다 native2ascii 명령어를 실행하여 파일을 다시 생성해 주어야 하는 번거로움이 있다. 그러나 ResourceBundleEditor 이클립스 플러그인을 사용하면 쉽게 작성할 수 있다

12.1.2 사용 방법

12.1.2.1 Java Util Method

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

properties 파일에 정의된 message 들을 GlueStaticContext.getResourceMessage(...)을 사용해서 가져올 수 있다.

getResourceMessage(String beanName, String key, Object[] arguments, Locale locale)

- beanName : ResourceBundleMessageSource 가 등록된 Bean ID
- key : property 파일에 정의된 message ID
- arguments : Message 에 바인딩 되는 값들
- locale : 적용 Locale, Locale 이 ko_KR 인 경우 message_ko_KR.properties 파일의 값을 읽어온다.

기타 형태

*getResourceMessage(key)

*getResourceMessage(key , locale)

*getResourceMessage(key, arguments)

*getResourceMessage(key, arguments, locale) :

Parameter 로 Locale 이 없을 경우 우선 순위

User 설정 Locale -> Glue Property 설정 Locale(default.ui.locale) -> System Locale

Parameter 로 beanName 이 없을 경우 Default 값 은 “messageSource.”이다

12.1.2.2 사용예:

<Java Code>

```
String key = "showtable.header.empinfo";
System.out.println("Default:::" + GlueStaticContext.getResourceMessage(key));
System.out.println("English:::" + GlueStaticContext.getResourceMessage(key,
Locale.US));
System.out.println("한국어:::" + GlueStaticContext.getResourceMessage(key,
Locale.KOREA));
```

<Jsp Code>

```
서버 기본값 : : : <%=GlueStaticContext.getResourceMessage(key)%><BR/>
English : : : <%=GlueStaticContext.getResourceMessage(key, Locale.US)%><BR/>
한국어 : : : <%=GlueStaticContext.getResourceMessage(key, Locale.KOREA)%><BR/>
```

12.2 Multi TimeZone

12.2.1 사용 방법

12.2.1.1 Java Util Method

getTimeZoneFormat(Date date, String timezoneid, String textformat)

- date : Text 형태로 변환 할 Date
- timezoneid : Text 형태로 변환 시 적용 할 TimeZone ID

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

- textformat : Text 형태로 변환 시 적용 될 Text Format
(Default 값 : yyyy-MM-dd HH:mm:ss (z Z))

getTimeZoneFormat(Date date, int offset, String textformat)

- date : Text 형태로 변환 할 Date
- offset : Text 형태로 변환 시 적용 할 TimeZone 의 OffSet 값
(분 단위 : 120 일 경우 GMT +02:00)
- textformat : Text 형태로 변환 시 적용 될 Text Format

기타 형태

getTimeZoneFormat(date, timezoneid)

getTimeZoneFormat(date, offset)

getTimeZoneFormat(date)

12.2.1.2 사용예:

<Java Code>

```
Date date = new Date();
String textformat = "yyyy-MM-dd HH:mm:ss (z Z)";
System.out.println("Default:::" +
GlueStaticContext.getTimeZoneFormat(date, TimeZone.getDefault().getID(),
textformat));
System.out.println("Asia/Seoul:::" +
GlueStaticContext.getTimeZoneFormat(date, "Asia/Seoul", textformat));
System.out.println("Canada/Eastern:::"
+GlueStaticContext.getTimeZoneFormat(date, "Canada/Eastern", textformat));
System.out.println("Asia/Shanghai:::"
+ GlueStaticContext.getTimeZoneFormat(date, "Asia/Shanghai", textformat));
System.out.println("+05:00:::" +GlueStaticContext.getTimeZoneFormat(date,
5*60, textformat));
System.out.println("-06:00:::" +GlueStaticContext.getTimeZoneFormat(date, -
6*60, textformat));
```

<Jsp Code>

```
서버 기본값 : : : <%=GlueStaticContext.getTimeZoneFormat(date,
TimeZone.getDefault().getID())%><BR/>
Asia/Seoul : : : <%=GlueStaticContext.getTimeZoneFormat(date,
"Asia/Seoul")%><BR/>
Europe/London : : : <%=GlueStaticContext.getTimeZoneFormat(date,
"Europe/London")%><BR/>
America/New_York : : : <%=GlueStaticContext.getTimeZoneFormat(date,
"America/New_York")%><BR/>
```

12.3 User Locale & TimeZone 설정

12.3.1 적용 예

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

12.3.1.1 Interceptors 설정

User Locale & TimeZone 을 설정한 Interceptors 를 dispatcher-servlet.xml 에 정의

```
<bean id="beanNameUrlMapping"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <value>
            /sample_login.mvc=controller
        </value>
    </property>
    <property name="interceptors">
        <list>
            <ref bean="sampleInterceptor"/>
        </list>
    </property>
</bean>
<bean id="sampleInterceptor" class="sample.CommonHandlerInterceptor" />
```

12.3.1.2 Interceptor 구현

Interceptor 에서 User Locale & TimeZone 정보를 Session 혹은 Cookies 에 저장한다

```
13 public class CommonHandlerInterceptor extends HandlerInterceptorAdapter {
14
15     @Override
16     public boolean preHandle(HttpServletRequest req, HttpServletResponse res, Object obj) throws Exception {
17
18         if(req.getParameter("userId") != null){
19             String local = req.getParameter("locale");
20             HttpSession session = req.getSession();
21
22             if(local.equals("Seoul")){
23                 session.setAttribute("user_locale", Locale.KOREAN);
24                 session.setAttribute("user_timezone", "Asia/Seoul");
25             }else if(local.equals("London")){
26                 session.setAttribute("user_locale", Locale.ENGLISH);
27                 session.setAttribute("user_timezone", "Europe/London");
28             }else if(local.equals("New_York")){
29                 session.setAttribute("user_locale", Locale.ENGLISH);
30                 session.setAttribute("user_timezone", "America/New_York");
31             }
32             return true;
33         }else{
34             res.sendRedirect(req.getContextPath()+"/login.jsp");
35             return false;
36         }
37     }
38 }
39
40 }
41
```

[그림 62] Interceptor 구현

12.3.1.3 JSP 에서 활용 예.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

Message : <%=GlueStaticContext.getResourceMessage(key,
(Locale)ctx.get("user_locale", GlueWebContext.SESSION_ATTR))%><BR/>
Time : <%=GlueStaticContext.getTimeZoneFormat(date,
(String)ctx.get("user_timezone", GlueWebContext.SESSION_ATTR))%><BR/>

```

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

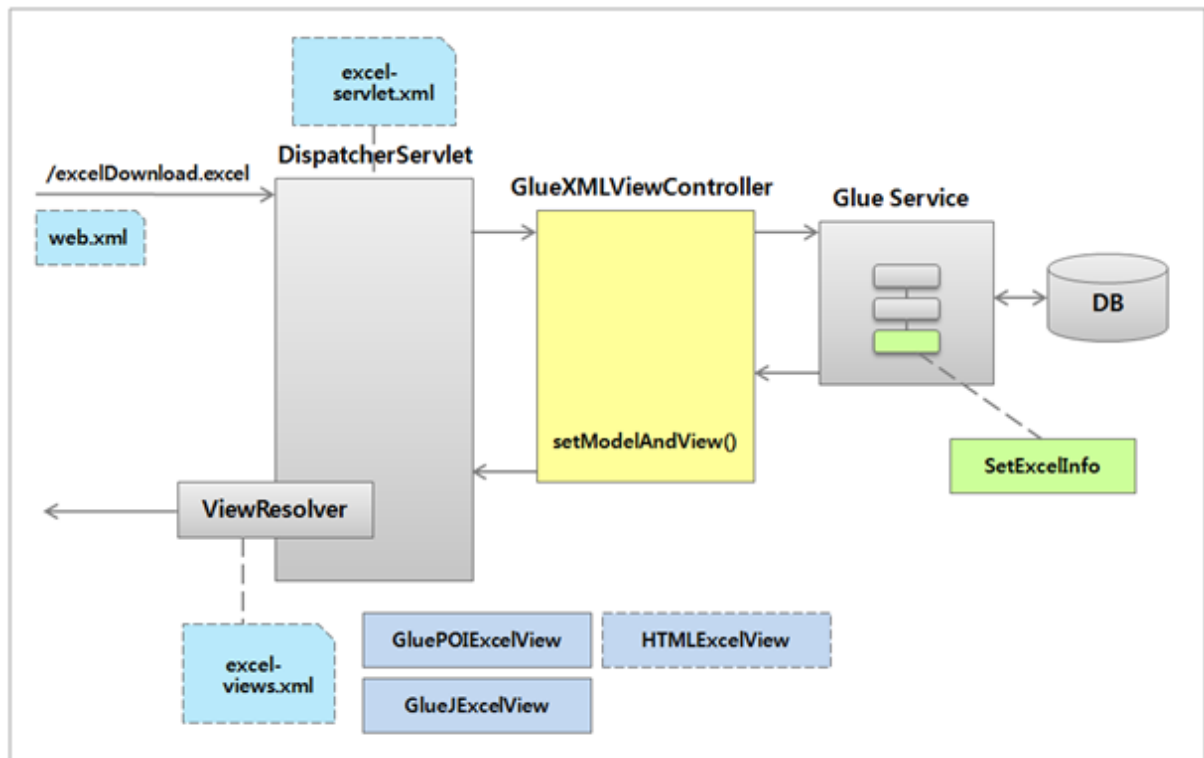
13. Excel

이번 장의 목적은 Glue Framework 기반으로 하여 Excel Export, Import 기능을 설명하는데 있다.

이번 장의 범위는 일반적인 Excel 형태에 대해서는 Template 형태로 기능을 지원하며 Site 별 필요한 추가 기능에 대해서는 개발 방법에 대한 가이드를 제공하는 것으로 한다.

13.1 Excel Export

Database 에서 조회한 Data 를 Excel 로 Export 하기 위해서는 View 객체를 정의하여 구현 가능하다. Glue Framework 에서는 아래 3 가지의 View Template 를 제공한다



[그림 63] Excel Export

13.1.1 GlueExcelInfo

Excel Export 를 위한 정보를 Set 하는 Activity Class 이다

13.1.2 GlueXMLViewController

Excel Import / Export 를 위한 Controller 로서 Excel 외 다양한 View 에 대해서도 Control 할 수 있다, 기본적으로 GlueSimpleController 와 동일하나 Request Parameter [viewname] 값에 해당하는 ModelAndView 를 XML 로부터 참조하여 View 를 만든다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

13.1.3 excel-views.xml

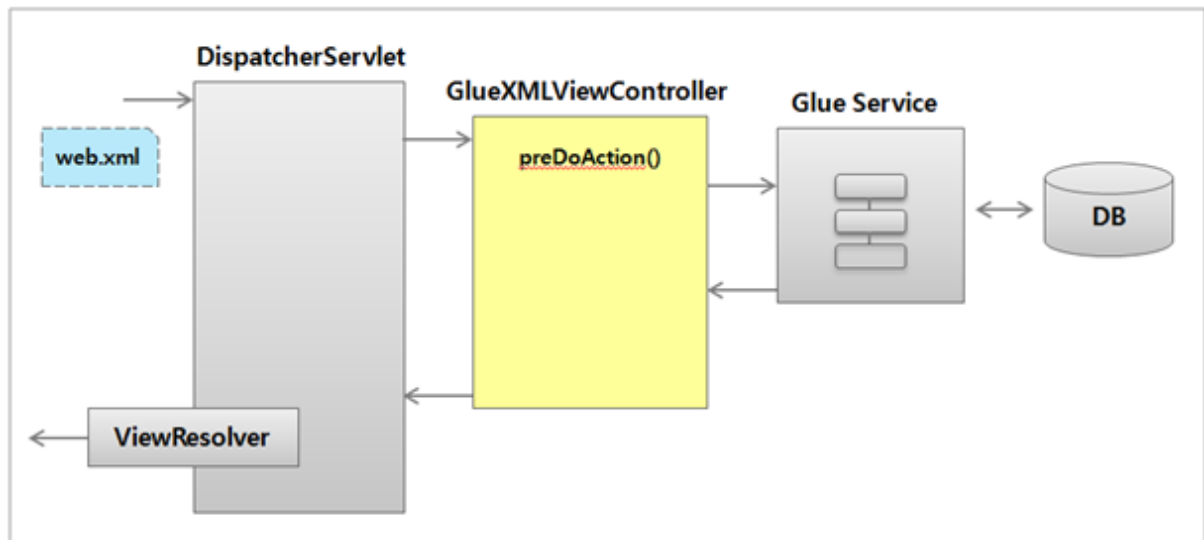
Excel Export 를 위한 ModelAndView 를 정의한 XML 파일이다.

이 xml 에서는 Excel Export 방식을 지정할 수 있으며 아래와 같이 3 가지 Tempate 을 제공한다

1. GluePOIExcelView : POI 의 HSSF 와 XSSF 을 구현하여 Excel 2007 이전 / 이후 버전을 모두 지원
2. GlueJExcelView : AbstractJExcelView 를 확장하여 구현하였으며 Excel 2003 이전 버전에 대해서만 지원 가능하다. 성능상으로 GluePOIExcelView 보다 빠르다.
3. HtmlExcelView - 구현 예정 : Html 로 디자인한 화면과 JSTL 을 이용하여 Template 으로 Excel 디자인 및 Format 을 다양하게 구현할 수 있다.

13.2 Excel Import

Database 로 Data 를 Insert 또는 Update 하기 위해 지정한 Excel 을 파싱하여 GlueContext 에 List<Map<String, Object>>형태로 저장하는 기능을 제공한다.



[그림 64] Excel Import

13.2.1 GlueXMLViewController

화면의 [multipart/form-data] Data 에 대해 MultipartRequest 객체를 생성한 후 Excel 파일을 파싱(Map 의 List 로 변환)하여 GlueContext 에 저장한다.

저장하는 Key 는 [sheet 명]이다.

예를 들어 Sheet 명이 emp 인 경우 emp sheet 의 Data 를 List<Map<String, Object>>형태로 변환하여 [emp]를 Key 로 GlueContext 에 저장한다.

13.2.2 Glue Service

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Glue AD 작성 시 GlueJDBCUpdate, GlueJDBCInsert, GlueJDBCDelete Activity 를 사용하여 Import 한 Data 를 DB 로 Update, Insert, Delete 가 가능하다

Property 작성 방법 : Property [chk-name]은 삭제하고 [list-key]에 Excel sheet 명을 입력.

13.3 Glue 환경 설정

Glue Framework 4.0 패키지에 포함된 설정파일을 기준으로 작성하였다

1. web.xml

기본적인 Glue Framework 4.x 버전에서 사용하는 web.xml 과 크게 다른 점은 없으며 Excel Control 을 위해 아래와 같은 DispatcherServlet 추가한다. url-pattern 를 이용해 *.excel 으로 URL 를 분리한다.

```
<servlet>
    <servlet-name>excel</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>excel</servlet-name>
    <url-pattern>*.excel</url-pattern>
</servlet-mapping>
```

2. {servlet-name}-servlet.xml

web.xml 에 servlet-name 을 excel 로 정의 했다면, excel-servlet.xml 을 아래와 같이 구성한다.

urlMapping 빈의 mappings property 에는 Excel 을 Upload, Download 하기 위한 URL 과 담당하는 Controller 명을 지정한다.

InternalResourceViewResolver 를 통해 XML 파일에 해당 View 가 존재하지 않는 경우 JSP 파일을 찾는다

```
<bean id="simpleUrlMapping"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <value>
            /**/excelUpload.excel=glueXMLViewController
            /**/excelDownload.excel=glueXMLViewController
        </value>
    </property>
</bean>
<!-- Excel Upload, Download Controller -->
<bean id="glueXMLViewController"
    class="com.poscoict.glueframework.web.control.spring.GlueXMLViewController" />
<!-- Excel View -->
<bean class="org.springframework.web.servlet.view.XmlViewResolver">
    <property name="location">
        <value>/WEB-INF/excel-views.xml</value>
    </property>
    <property name="order" value="0" />
</bean>
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/excel/" />
    <property name="suffix" value=".jsp" />
</bean>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

3. /WEB-INF/excel-views.xml

XMLViewResolver 의 location property 에 정의된 파일은 다음과 같이 구성한다.

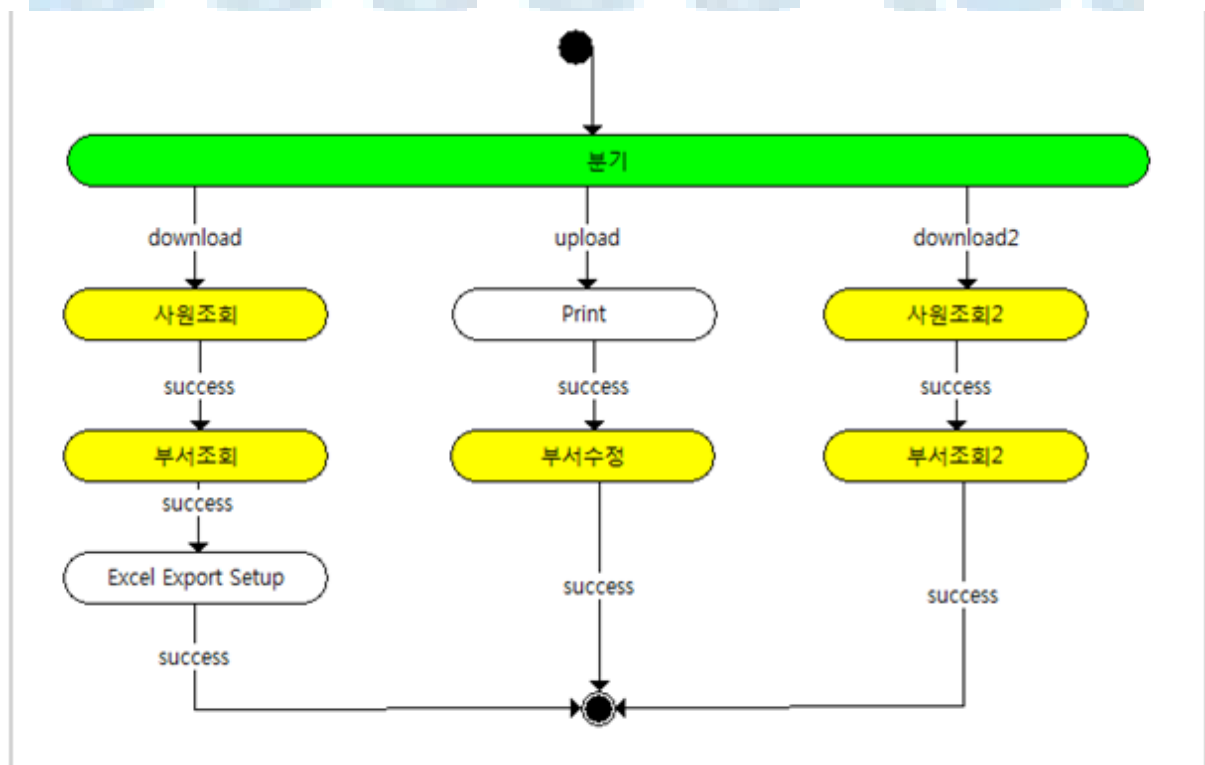
```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
  <!-- Generic View for excel included 2007 higher -->
  <bean id="GluePOIExcelView"
    class="com.poscoict.glueframework.web.control.spring.view.GluePOIExcelView"/>
  <!-- Generic View for JExcel(JXL) -->
  <bean id="GlueJExcelView"
    class="com.poscoict.glueframework.web.control.spring.view.GlueJExcelView"/>
</beans>
```

bean id 는 화면에서는 사용할 Model 명을 정의한다.

13.4 Sample 화면

다음 ActivityDiagram 는 아래 3 가지 Case 를 다루고 있다

1. download : 사원 및 부서 테이블을 조회한 후 Excel Export 를 위한 정보를 Set 하는 Activity 로 구성된 Case
2. download2 : 사원 및 부서 테이블 조회만 수행하며 Excel Export 를 위한 정보는 화면(JSP)에서 받는 Case
3. upload : Upload 된 Excel Data 로 Update 하는 Case



[그림 65] excel-service.xml

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

아래는 Download 를 위한 JSP 페이지의 일부이다. `viewname` 으로 `GluePOIExcelView` 를 지정하여 POI Library 를 이용하여 Excel 을 Generate 하며 `GlueJExcelView` 나 개발자가 직접 구현한 view 명을 지정하여 사용할 수 있다.

```
<form name="downloadForm" action="./excelDownload.excel" method="post">
<input type="hidden" name="ServiceName" value="excel-service">
<input type="hidden" name="download" value="1">
<input type="hidden" name="viewname" value="GluePOIExcelView">
<input type="text" name="deptno" value="20">
<input type="submit" name="Excel Download">
</form>
```

`GlueExcelInfo` Activity 에는 Excel 을 생성하기 위한 부가 정보를 입력한다. 작성예는 아래와 같고 자세한 사용법은 `GlueExcelInfo` JavaDoc 을 참고한다

```
<activity name="Excel Export Setup"
  class="com.poscoict.glueframework.biz.activity.GlueExcelInfo">
  <transition name="success" value="end" />
  <property name="sheet-name" value="사원정보, 부서정보" />
  <property name="dept-column-width" value="80,120" />
  <property name="dept-column-id" value="DEPTNO,DNAME" />
  <property name="sheet-id" value="emp,dept" />
  <property name="dept-column-name" value="부서번호, 부서명" />
</activity>
```

아래는 Excel Import 를 위한 JSP 페이지의 일부이다. `forwardname` 으로는 glue 수행 후 jsp 페이지명을 지정할 수 있다.

```
<form name="uploadForm" action="./excelUpload.excel" method="post" onSubmit="return
goUploadExcel();" enctype="multipart/form-data">
<input type="file" name="excelFileName" />
<input type="hidden" name="ServiceName" value="excel-service">
<input type="hidden" name="upload" value="1">
<input type="hidden" name="forwardname" value="excelUpload">
<input type="submit" ></button>
</form>
```

Excel Import 와 관련 파싱된 Excel 은 `GlueContext` 에 저장되며 `Reusable Activity` 를 활용하여 수정, 삭제, 등록이 가능하다. 아래는 그 예이다.

```
<activity name="부서수정"
  class="com.poscoict.glueframework.biz.activity.GlueJdbcModify">
  <transition name="success" value="end" />
  <property name="result-key" value="dept.update_resultList" />
  <property name="param-bindings" value="DNAME=DNAME|DEPTNO=DEPTNO" />
  <property name="sql-key" value="dept.update" />
  <property name="dao" value="test-dao" />
  <property name="list-key" value="부서정보" />
</activity>
```

13.5 Custumizing

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Glue 에서 제공하는 View Class 는 일반적인 Case 에 대해 제공하며 각 Site 별 추가 요구사항이 존재하는 경우 별도의 Excel View 를 만들도록 한다.

13.5.1 POI 용 View 제작

POI API 에서 CellStyle 을 생성하기 위해서는 workbook 이 필요하기 때문에 아래와 같이 View Class 의 getExcelStyle()을 재정의하도록 한다. 아래는 숫자 타입의 Cell 에 대해 왼쪽 정렬하는 샘플 소스이다.

```
public class LeftAlignExcelView extends GluePOIExcelView {
    . . . . .
    protected ExcelStyle getExcelStyle(Workbook workbook){
        ExcelStyle excelStyle = new ExcelStyle(workbook);
        // excel style 을 수정하는 로직
        CellStyle bigDecimalCellStyle =excelStyle.getBigDecimalCellStyle();
        bigDecimalCellStyle.setAlign(CellStyle.ALIGN_LEFT);
        . . . . .
        return excelStyle;
    }
}
```

13.5.2 JExcel View 제작

GlueJExcelView 에서는 GlueContext 에서 [excel-style]로 지정된 객체를 Excel Style 로 사용 가능하다. 따라서 Excel Style 을 변경하고자 한다면 JExcelFormat Class 를 확장하여 Customizing 후 Activity Diagram 에서 GlueContext 에 [excel-style]를 Key 로 저장하면 된다.

```
public class JExcelFormat {
    WritableCellFormat headerCellFormat;        //
    WritableCellFormat bigDecimalCellFormat;
    WritableCellFormat timestampCellFormat;
    WritableCellFormat stringCellFormat;
    String dateFormat = "yyyy-mm-dd";
    . . . . .
}
```

와 같이 정의된 JExcelFormat 을 상속받아

```
public class MyExcelFormat extends JExcelFormat {
    public void setHeaderCellFormat(WritableCellFormat headerCellFormat) {
        this.headerCellFormat = headerCellFormat;
        this.headerCellFormat.setBorder(Border.ALL, BorderLineStyle.THICK);
        . . . . .
    }
    . . . . .
}
```

와 같이 원하는 부분을 수정한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

14. 웹서비스

이 장은 JAX-WS, RESTful 관련 Glue Framework 과의 관계와 연결 방법에 대해 설명하는 것을 목적으로 한다. 각 단원 마다 예제 및 Sample Code 를 삽입하여 개발자가 쉽게 따라 개발할 수 있도록 하였다.

Glue Framework 를 사용하여 웹 서비스를 개발할 경우 필요한 내용들을 설명한다. 본 문서에서는 JAX-WS, RESTful 관련 개념이나 어노테이션(annotation) 관련 부분은 꼭 필요한 부분만 설명하고 좀 더 구체적인 내용은 사이트를 참고하도록 한다.

14.1 JAX-WS

JAX-WS(Java API for XML Web Services)는 웹 서비스를 생성하는 자바 API 로써, Java EE 의 일부이다.

다른 Java EE 의 자바 API 와 같이, JAX-WS 는 Java SE 5 에서 도입된 어노테이션(annotation)을 사용하여 웹 서비스 클라이언트 및 서버 모듈의 개발 및 배포를 쉽게 하고 있다. JAX-WS 는 JAX-RPC 표준을 발전 시킨 개념으로 XML 의 바인딩을 위한 JAXB 표준과 표준 스트리밍 파서를 위한 StAX 표준, 기능이 향상된 새로운 SAAJ 표준을 기반으로 통합, 발전된 자바 진영의 노력의 산물이다.

JAX-WS 2.0 이 JAX-RPC 1.1 의 차후 버전으로 JAX-RPC 1.1 의 다음 버전인 JAX-RPC 2.0 이 나오면서, PC 라는 용어 대신 메시지 지향 웹 서비스인 WS 로 대체되어 JAX-WS 2.0 으로 불리게 되었다.

- JAX-WS : Java API for XML-Based Web Services
- JAX-RPC : Java API for XML-Based RPC(Remote Procedure call)

많이 사용되는 웹 서비스 오픈 소스 프레임워크로는 Apache CXF, Apache Axis/ Axis2, Spring Web Services 등이 존재하며 여러 프레임워크 중 사용하고자 하는 목적에 적합한 웹 서비스 프레임워크를 선정해야하는데 Glue Framework 에서는 Apache CXF 를 선정하였다.

Apache CXF 의 장점은 다음과 같다.

1. JAX-WS 지원: CXF 에서는 JAX-WS API 를 구현하고 있어서 웹 서비스 구현을 쉽게 하고 있다.
2. Spring Integration: CXF 는 Spring 2.X 이상을 지원하며 endpoint 설정이나 client injection 등 Spring 과의 통합을 용이 하게 한다.
3. Apache Licensed: 아파치 라이선스 사용으로 활용에 자유롭다.

JAXB 는 Java Architecture for XML Binding 의 약자로 XML Schema 로부터 클래스 데이터를 바인딩하여 XML 로부터 객체를 Unmarshalling 하거나, 반대로 객체를 XML 로 저장하는 Marshalling 을 수행할 수 있도록 해 준다. (JAXB Databinding) 이러한 JAXB 의 특징은 다음과 같다.

1. Apache CXF 사용 시 디폴트 Databinding 방식이 JAXB 이므로, JAXB 사용을 위한 추가 설정이 필요없다.
2. JavaBeans 코드에 @XmlElement 와 같은 Annotation 을 설정하여 element 명을 변경시킬 수 있다.
3. JavaBeans 코드에 Annotation 설정 없이도 JAXB 를 사용할 수 있다.
4. List 형태의 타입은 지원하나 Map 형태의 타입에 대해서는 Databinding 을 기본적으로 지원하지 않으므로 Map 형태의 타입을 이용하여 Databinding 하기 위해서는 XmlAdapter 클래스를 추가로 작성해줘야 한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

JAXB 사용시 SEI (Service Endpoint Interface) 클래스에 정의된 메소드의 입력값이나 리턴값으로 참조되지 않은 Java Type 클래스가 사용될 경우, Runtime 시 Databinding 이 일어날 때 다음과 같은 예외 메시지가 나오게 된다.

```
org.apache.cxf.interceptor.Fault: Marshalling Error: xxxxxxxx nor any of its super class is known to this context.
```

이 경우 아래와 같은 방법을 통해서 해결 할 수 있다.

SEI(Service Endpoint Interface) 클래스의 메소드에는 List 형태의 결과값을 리턴하는 메소드가 존재하고 해당 List 내에 저장된 Java Type 클래스에 대해서는 참조하는 메소드가 없는 경우, 클라이언트가 해당 메소드를 호출할 때 Runtime 시에 JAXB Databinding 을 시도하려고 하면서 Marshalling Error 가 발생하게 된다. 이를 방지하기 위해서 @XmlSeeAlso Annotation 혹은 JDK1.5 의 Generic Type 을 설정하도록 한다.

다음은 @XmlSeeAlso 사용 예제이다.

```
@WebService
@XmlSeeAlso({EmpVo.class})
public interface JaxService {
    public List getList( . . . );
    . . .
```

다음은 JDK 1.5 의 Generic Type 사용 예제이다.

```
@WebService
public interface JaxService {
    public List< EmpVo> getList ( . . . );
    ...
```

14.1.1 Web Service 작성

Web Service 를 구현할 때는 인터페이스 와 구현 클래스로 작성한다.

인터페이스(interface)는 상단에 @WebService Annotation 을 작성해줘야 한다. @WebService annotation 의 속성 값들도 다양하게 존재하고, 각 메소드와 파라미터 별로 정의할 수 있는 Annotation 의 종류도 여러가지가 있으나 필수 사항으로 필요한 Annotation 은 @WebService 뿐이다.

```
@WebService
public interface JaxSampleService {
    public String findString(String id) throws Exception;
    public HashMap<String,Object> findMap(String id) throws Exception;
    public List<EmpVO> findList() throws Exception;
    public List<EmpVO> doGlueService(HashMap<String,Object> glueMap) throws Exception;
}
```

인터페이스(interface) 에서 @WebService 선언해 주었으므로 구현 클래스에서는 Web Service 구현과 관련된 부분 없이 작성될 수 있다. 인터페이스(interface)에서 설정하지 않은 부분을 추가로 설정해야 할 경우에는 구현 클래스 상단에 @WebService Annotation 관련해서 추가 작성해줄 수도 있다. 아래 예제에서는 인터페이스 클래스 상단에 @WebService Annotation 을 작성해주었으므로 생략했다.

```
public class JaxSampleServiceImpl implements JaxSampleService
{
    . . .
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

@Override
public List<EmpVO> doGlueService(HashMap<String, Object> glueMap)
    throws Exception
{
    GlueBizController bizIf = GlueBizProvider.getController();
    GlueContext ctx = new GlueDefaultContext();
    ctx.putAll(glueMap);
    bizIf.doAction(ctx);
    List<EmpVO> rowSet = (List<EmpVO>)ctx.get("EmpList");
    return rowSet;
}
. . .
}

```

14.1.2 Server 부분

작성된 서비스를 Web Service 로 노출시키는 서버를 구동하기 위해서는 여러가지 방식이 지원되는데 이중 Spring Configuration XML - <jaxws:endpoint/> tag 를 사용하여 서버를 구동하는 방법에 대해서 살펴보자. <jaxws:endpoint/> tag 의 필수 속성값에 대한 설명은 다음 표와 같다. 아래 표에 나와있지 않은 속성들도 여러 가지가 존재한다.

구분	설명
id	spring bean id 를 작성한다
implementor	구현 클래스를 작성한다. 클래스명 대신에 spring bean id 로 대체하고자 하면 bean id 앞에 #을 붙여서 작성하면 된다.
address	서비스가 동작할 주소를 상대 경로로 작성한다.

[표 17] <jaxws:endpoint/> tag 의 필수 속성값

아래는 <jaxws:endpoint/> tag 속성을 정의한 xml 의 일부이다

```

<!-- Load CXF modules from cxf.jar -->
<importresource="classpath:META-INF/cxf/cxf.xml"/>
<importresource="classpath:META-INF/cxf/cxf-extension-soap.xml"/>
<importresource="classpath:META-INF/cxf/cxf-servlet.xml"/>
<beanid="jaxSampleServiceImpl"class="sample.jax.JaxSampleServiceImpl"></bean>
<jaxws:endpointid="jaxSampleService"
    implementor="#jaxSampleServiceImpl"
    address="/jaxSample"/>

```

일반적으로 Tomcat 서버 등의 WAS 를 이용하여 웹 어플리케이션을 구동하고 비즈니스 레이어의 서비스를 Web Service 로 노출시키게 되는데, 이때 web.xml 파일에 Spring 속성 정의 XML 파일을 org.springframework.web.context.ContextLoaderListener 를 이용하여 등록시켜 줘야 한다. 다음은 CXFServlet 과 ContextLoaderListener 를 정의한 web.xml 의 일부이다.

```

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>WEB-INF/jaxws-server-endpoint.xml</param-value>
</context-param>
<listener>
    <listener-class>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
<servlet>
    <servlet-name>CXFServlet</servlet-name>
    <display-name>CXF Servlet</display-name>
    <servlet-class>
        org.apache.cxf.transport.servlet.CXFServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>CXFServlet</servlet-name>
    <url-pattern>/ws/*</url-pattern>
</servlet-mapping>

```

14.1.3 Client 부분

<jaxws:client/> tag 의 각각의 Attribute 속성값에 대한 설명은 다음 표와 같다. 아래 표에 나와있지 않은 속성들도 여러 가지가 존재한다. 단, 여기서는 필수적으로 작성해야 하는 Attribute 속성값에 대한 설명을 작성해놓은 것으로 나머지 속성 값들에 대한 설명은 [JAX-WS Spring Configuration] 을 참고하도록 한다.

구분	설명
id	spring bean id 를 작성한다.
serviceBean	서비스의 인터페이스 클래스를 작성한다.
address	서비스 접근 URL Address 를 절대 경로로 작성한다.

다음은 Web Services 로 노출된 Movie Service 에 접근하는 <jaxws:client/> tag 속성을 정의한 xml 의 일부이다.

```

<jaxws:client id="glueJaxWsClientService"
    serviceClass="com.poscoict.glueframework.jaxws.GlueJaxWebService"
    address="http://localhost:8080/sample/ws/glueJaxWs" />

```

다음은 앞서 정의한 속성 설정 파일들을 기반으로 하여 Web Services 로 노출된 GlueJaxWebService 에 접근하는 java 소스의 일부이다.

```

GlueJaxWebService sampleService
= (GlueJaxWebService) GlueStaticContext.getBeanFactory()
    .getBeanObject("glueJaxWsClientService");
try{
    HashMap<String,Object> glueMap = new HashMap<String, Object>();
    glueMap.put("ServiceName", "sample01-service");
    glueMap.put(GlueBizControlConstants.JAXWS_RESULT, "EmpList");
    glueMap.put("find", "event");
    glueMap.put("deptno", "10");
    List<GlueRowImpl> resultList
        = (List<GlueRowImpl>) sampleService.doGlueServiceGetList(glueMap);
    . . .
}

```

	문서명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Apache CXF 에서 제공하는 JaxWsProxyFactoryBean 를 직접 이용하여 Web Services 로 노출된 Service 에 접근하는 것도 가능하며, java 소스 일부는 다음과 같다.

```
JaxWsProxyFactoryBean factory = new JaxWsProxyFactoryBean();
factory.setServiceClass(GlueJaxWebService.class);

factory.setAddress("http://localhost:8080/sample/ws/glueJaxWs");
GlueJaxWebService sampleService = (GlueJaxWebService) factory.create();
```

14.1.4 Glue 에서 제공하는 WebService 클래스

glue-ws 모듈을 통해 다음 2 가지 형태의 WebService 를 제공한다.

- GlueJaxWebService interface: jax-ws 기반에서 웹 서비스 기능을 제공하기 위한 interface 로 Glue Service 를 실행 시키고 GlueContext 에서 해당 객체를 리턴하는 메서드들을 제공한다. 파라미터나 리턴 객체가 Map 이나 List 인 경우 map 이나 list Type 의 객체 혹은 interface 나 abstract Class 가 담길 경우 JAXB Databinding 에서 Error 가 발생하므로 맵핑 객체에 주의해야 한다.
- GlueJaxWebServiceImpl 구현체: jax-ws 기반에서 웹 서비스 기능을 제공하기 위한 GlueJaxWebService 의 구현체
- GlueWebServiceJson interface: jax-ws 기반에서 웹 서비스 기능을 제공하기 위한 interface 로 Glue Service 를 실행 시키고 GlueContext 에서 해당 객체를 리턴하는 메서드들을 제공한다. 파라미터와 리턴 String 을 JSON 형태로 제공한다.
- GlueWebServiceJsonImpl 구현체: jax-ws 기반에서 웹 서비스 기능을 제공하기 위한 GlueWebServiceJson 의 구현체

구분	메소드	설명
GlueJaxWebService	doGlueServiceGetStringByName	Glue Service 를 실행 시키고 GlueContext 에서 String 객체를 Return 한다.
	doGlueServiceGetListByName	Glue Service 를 실행 시키고 GlueContext 에서 List<?>객체를 Return 한다.
	doGlueServiceGetMapByName	Glue Service 를 실행 시키고 GlueContext 에서 HashMap<String,Object>객체를 Return 한다.
	doGlueServiceGetString	Glue Service 를 실행 시키고 GlueContext 에서 String 객체를 Return 한다.
	doGlueServiceGetList	Glue Service 를 실행 시키고 GlueContext 에서 List<?>객체를 Return 한다.
	doGlueServiceGetMap	Glue Service 를 실행 시키고 GlueContext 에서 HashMap<String,Object>객체를 Return 한다.
GlueWebServiceJson	doGlueServiceByJson	Glue Service 를 실행 시키고 GlueContext 에서 Json 형태의 String 객체를 Return 한다.

	문서명	Glue Framework 개발자 가이드
	제품버전	V 4.0

14.2 RESTful

REST란 ROA(Resource Oriented Architecture)를 따르는 웹 서비스 디자인 표준으로 웹의 모든 리소스를 URI로 표현하고, 이를 구조적이고 유기적으로 연결하여, 비 상태 지향적인 방법으로, 일관된 method를 사용하여 리소스를 사용한다.

REST 방식의 웹서비스는 잘 정의된 Cool URI로 리소스를 표현한다는 특징을 갖는다.

무분별한 파라미터의 남발이 아니라, 마치 오브젝트의 멤버변수를 따라가듯이 예를 들면 아래와 같다.

```
http://www.glue.net/user/mk/age/32
```

기존의 서블릿을 이용한 URI는 아래와 같은 방식이었다.

```
http://www.glue.net/finduser.jsp?user=mk&age=32
```

일반적으로 ROA는

- 웹의 모든 리소스를 URI로 표현하고
- 모든 리소스를 구조적이고 유기적으로 연결하여
- 비 상태 지향적인 방법으로
- 정해진 method만을 사용하여 리소스를 사용하는 아키텍처

라고 4가지로 정의한다.

ROA를 따르는 RESTful 웹서비스는 아래와 같은 4가지 속성을 갖는다.

1. Addressability (주소로 표현 가능함): 제공하는 모든 정보를 URI로 표시할 수 있어야 한다. 직접 URI로 접근할 수 없고 HyperLink를 따라서만 해당 리소스에 접근할 수 있다면 이는 RESTful하지 않은 웹서비스이다.
2. Connectedness (연결됨): 일반 웹 페이지처럼 하나의 리소스들은 서로 주변의 연관 리소스들과 연결되어 표현(Presentation)되어야 한다.

다음은 독립적인 리소스 예제이며,

```
<user>
<name>MK</name>
</user>
```

다음은 관련 리소스(home, office)가 잘 연결된 리소스 예제이다.

```
<user>
<name>MK</name>
<home>MK/home/</home>
<office>MK/office</office>
</user>
```

3. Statelessness (상태 없음): 현재 클라이언트의 상태를 절대로 서버에서 관리하지 않아야 한다. 모든 요청은 일회성의 성격을 가지며 이전의 요청에 영향을 받지 말아야 한다. 세션을 유지하지 않기 때문에 서버 로드 발란싱이 매우 유리하다.
4. Homogeneous Interface (동일한 인터페이스): HTTP에서 제공하는 기본적인 4가지의 method와 추가적인 2가지의 method를 이용해서 리소스의 모든 동작을 정의한다.

구분	method	추가설명
리소스 조회	GET	

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

새로운 리소스 생성	PUT, POST	새로운 리소스의 URI 를 생성하는 주체가 서버이면 POST 를 사용
존재하는 리소스 변경	PUT	
존재하는 리소스 삭제	DELETE	
존재하는 리소스 메타데이터 보기	HEAD	
존재하는 리소스의 지원 method 체크	OPTION	

14.2.1 Server 파트

Spring 기반에서 RESTful 방식으로 구현한다

Spring 의 REST 를 위한 기능은 모두 Spring MVC 를 기반으로 지원된다. 다양한 Annotation 과 HTTP Request/Response Body 메시지 처리를 위한 `HttpMessageConverter`, Content Negotiation 지원을 위한 `ViewResolver`, 모든 HTTP method 사용을 위한 `Filter`, 그리고 REST 클라이언트 어플리케이션 개발에 도움을 주는 `RestTemplate` 등이 있다.

`web.xml` 에 아래와 같이 확장자로 맵핑되지 않도록 `DispatcherServlet` 을 등록한다.

```
<servlet>
  <servlet-name>restful</servlet-name>
  <servlet-class>
    org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>restful</servlet-name>
  <url-pattern>/restful/*</url-pattern>
</servlet-mapping>
```

`DispatcherServlet` 은 `{servlet-name}-servlet.xml` 을 필요로 하며, `web.xml` 이 위와 같다면 `restful-servlet.xml` 을 필요로 한다. 그 내용은 다음과 같이 설정한다.

```
<context:component-scan base-package="sample.controller" />
<mvc:annotation-driven />
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix" value="/WEB-INF/pages/" />
  <property name="suffix" value=".jsp" />
</bean>
<bean name="ViewResolver"
class="org.springframework.web.servlet.view.XmlViewResolver">
  <property name="order" value="1"/>
  <property name="location" value="/WEB-INF/views.xml"/>
</bean>
```

json 으로 데이터를 주고 받을 경우 json 데이터를 변환해줄 `AnnotationMethodHandlerAdater` 를 빈으로 등록하고 `messageConverters` 속성에 `Json 용 Converters` 클래스를 추가해야 하는데 `<mvc:annotation-driven />`이 해당 bean 설정을 자동으로 설정해 준다. 해당 빈 설정을 변경해야 할 경우에는 `<mvc:annotation-driven />`을 사용하지 말고 `AnnotationMethodHandlerAdater` 등의 Bean 을 직접 등록해주면 된다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

@controller 가 있는 Class 를 자동 스캔 하기 위해서 <context:component-scan base-package=...>을 통해서 해당 package 를 설정해 주어야 한다. 그래서 <context:component-scan base-package="sample.controller" />이 사용되었다.

ViewResolver 에서는 views.xml 을 필요로 하며, MappingJacksonJsonView 를 사용할 수 있도록 해당 Bean 을 다음과 같이 등록해야 한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
<bean name="jsonView"
class=org.springframework.web.servlet.view.json.MappingJacksonJsonView"/>
</beans>
```

<context:component-scan base-package="sample.controller" /> 를 통해 자동 스캔되는 Class 는 다음과 같은 형태이다.

```
@Controller
public class RestfulController
{
    /** logger */
    protected GlueLog logger = GlueLogFactory.getLogger(getClass());
    @RequestMapping(value = "/glue/{serviceName}", method = RequestMethod.GET)
    public ModelAndView doGlueServiceGet(@PathVariable String serviceName)
    {
        ModelAndView mav = new ModelAndView("jsonView");
        //비지니스 로직 구현
        return mav;
    }
}
```

@Controller 어노테이션은 해당 클래스가 컨트롤러의 역할을 한다는 것을 나타낸다. 스프링에서는 어떤 컨트롤러 기반 클래스도 확장할 필요가 없고 서블릿 API 를 참조할 필요도 없다. 하지만 필요하다면 서블릿에 특화된 기능을 참조할 수 있다.

@Controller 어노테이션은 어노테이션이 붙은 클래스의 스테레오 타입처럼 동작하고 클래스의 역할을 나타낸다. 디스패처는 매핑된 메서드에 이러한 어노테이션이 붙은 클래스를 찾고 @RequestMapping 어노테이션을 탐지한다.

디스패처의 컨텍스트에 표준 스프링 빈 정의를 사용해서 어노테이션이 붙은 컨트롤러 빈을 명시적으로 정의할 수 있다. 하지만 클래스패스에서 컴포넌트 클래스를 탐지와 컴포넌트 클래스들을 위해 빈 정의의 자동등록에 대한 스프링의 일반적인 지원에 맞추어 @Controller 스테레오타입도 자동탐지(<context:component-scan base-...>)가 가능하다.

특정 URL 을 매핑하려면 전체 클래스나 특정 핸들러 메서드에 @RequestMapping 어노테이션을 사용해야 한다. 보통 클래스수준의 어노테이션은 폼(form) 컨트롤러에 특정 요청 경로(또는 경로 패턴)을 매핑하고 추가적인 메서드 수준의 어노테이션은 특정 HTTP 요청 메서드("GET", "POST" 등)나 HTTP 요청 파라미터 상태로 매핑 범위를 좁힌다.

```
@Controller
@RequestMapping("/poscoict")
public class AppointmentsController
{
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

@RequestMapping(method = RequestMethod.GET)
public ModelAndView()
{
    // 구현
}
@RequestMapping(value="/test", method = RequestMethod.GET)
public ModelAndViewNew()
{
    // 구현
}
}

```

위 예제에서 `@RequestMapping` 를 여러 곳에서 사용했다. 먼저 클래스 수준에서 사용했는데 이것은 컨트롤러의 모든 핸들링 메서드는 `/poscoict` 경로에 상대적이라는 것을 나타낸다. `get()` 메서드의 `@RequestMapping` 의 `method` 만을 정의했는데, 이것은 GET 요청만 받아들인다는 것을 의미한다. 즉, `/poscoict` 에 대한 HTTP GET 요청만 이 메서드를 실행한다. POST 요청 처리를 위해서도 마찬가지로 방식으로 구현할 수 있다. `getNew()` 은 HTTP 메서드와 경로를 함께 사용했으므로 `poscoict/new` 에 대한 GET 요청을 이 메서드가 처리한다.

Method 정의시 일부 오래된 장비나 서버의 경우 GET, POST 만 허용하는 경우도 있으므로 주의해야 한다.

클래스 수준의 `@RequestMapping` 는 필수가 아니다. 클래스에 `@RequestMapping` 를 사용하지 않으면 모든 경로는 상대경로가 아니라 절대경로가 된다.

스프링 MVC 에서 URI 템플릿 변수의 값에 바인딩하려고 메서드 인자에 `@PathVariable` 어노테이션을 사용할 수 있다.

```

@RequestMapping(value="/glue/{loginid}", method=RequestMethod.GET)
public String findOwner(@PathVariable("loginid") String id) {
    // 구현
}

```

`@PathVariable` 어노테이션을 처리하려면 스프링 MVC 가 이름과 일치하는 URI 템플릿 변수(`loginid`)를 찾아야 한다. 어노테이션에서 이를 지정할 수 있다.

또는 URI 템플릿 변수 이름이 메서드 인자의 이름과 일치한다면 생략할 수 있다.

메서드는 다수의 `@PathVariable` 어노테이션을 가질 수 있다.

`@PathVariable` 인자는 `int`, `long`, `Date` 등의 간단한 어떠한 타입이라도 될 수 있다. 스프링이 자동으로 적절한 타입으로 변환하거나 타입변환에 실패한다면 `TypeMismatchException` 를 던진다.

14.2.2 Client 파트

다음은 jquery 를 이용한 호출 예제이다.

```

$.ajax({
    type: 'GET',
    url: "http://localhost:8080/GlueSample/restful/glue/sample01-service/",
    data: formdata,
    success: function(resultdata){
        //비즈니스 로직 구현
        // resultdata 은 해당 웹서비스에서 넘어온 값
    }
});

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

다음은 RestTemplate 을 사용한 예제이다.

```
RestTemplate restTemplate = new RestTemplate();
String result = restTemplate.getForObject
( "http://localhost:8080/GlueSample/restful/glue/sample01-service/" , String.class );
```

위와 같이 가급적 RestTemplate 클래스의 빈으로 등록하여 사용하는 것을 권장한다.

GET 방식의 요청일 경우에는 getForObject(String, Class, String...)을 사용하고 POST 방식일 경우에는 postForLocation(String, Object, String...) , PUT 방식일 경우에는 put(String, Object, String...),DELETE 방식일 경우에는 delete(String, String...)를 사용한다.

14.2.3 Glue 에서 제공하는 RESTful 용 Controller

Glue 에서는 특정 서비스를 실행 시키고 결과 값을 JSON 형태로 반환하는 RestfulController 를 제공하고 있다.

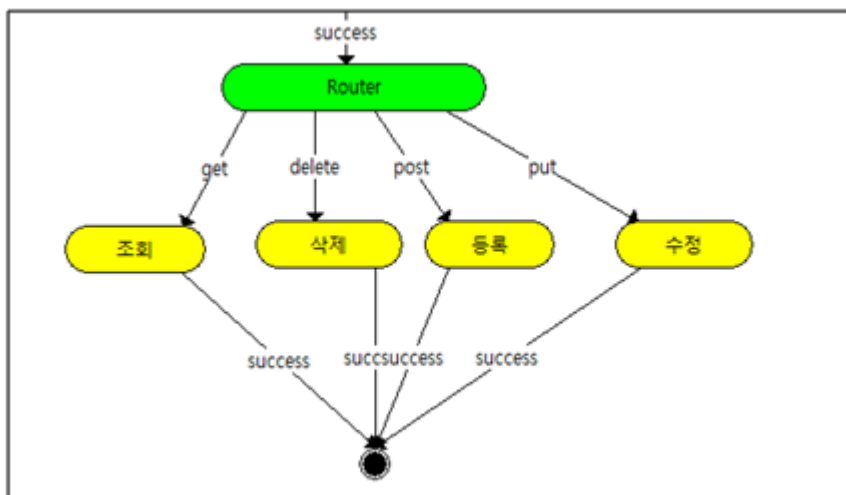
servlet.xml 에 Controller 스캔 Package 를 다음과 같이 정의해준다.

```
<context:component-scan
    base-package="com.poscoict.glueframework.web.control.restful" />
```

Controller 는 다음과 같은 URI 패턴을 처리할 수 있다.

- /glue/{serviceName} : URI 의 serviceName 과 Service 명이 일치하는 Glue Service 가 실행된다.
- /glue/{serviceName}/data/{gluedata} : URI 의 serviceName 과 Service 명이 일치하는 Glue Service 가 실행되며 URI 의 gluedata 값은 "gluedata"를 Key 로 GlueContext 에 저장된다.

RequestMethod 로는 GET,POST,PUT,DELETE 방식이 지원되며 어떤 방식의 요청이었는지는 "action"을 Key 로 GlueContext 에 소문자로 저장된다. 해당 요청이 Key 로도 등록되므로 아래와 같이 Default Router 로 분기도 가능하다.



[그림 66] Default Router 분기

POST 방식은 RequestParam 으로 넘어온 값을 모두 GlueContext 에 전달하며, PUT 방식의 경우에는 RequestBody 로 넘어온 값을 모두 GlueContext 에 전달한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

Glue 에서 제공하는 RESTful 용 Controller 가 사용되었다면, Client 부분에서는 ajax 방식이나 RestTemplate 을 사용하여 서비스를 호출 할 수 있으면 JSON 형식의 데이터가 Return 된다.

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

15. Scheduler

이 장의 목적은 Scheduler 관련 Framework 의 관계와 연결방법에 대해 설명하는 것으로 한다. 각 단위마다 예제 및 Sample Code 를 삽입하여 개발자가 쉽게 따라 개발할 수 있도록 하였다.

Glue Framework 를 사용하여 Scheduler 를 개발할 경우 필요한 내용들을 설명한다. 본 문서에서는 Spring Scheduler 나 Quartz 관련 부분은 꼭 필요한 부분만 설명하고 좀 더 구체적인 내용은 사이트를 참고하도록 한다.

Job Scheduler 라 함은 특정 프로그램 모듈(Job)을 정주기로 또는 이벤트성으로 실행시켜 주는 시스템을 의미한다.

Glue Framework 에서는 오픈소스인 Spring Scheduler 혹은 Quartz Job Scheduler 를 이용하여 스케줄러 기능을 제공하며, 단순한 Job 부터 복잡한 Job 까지 모두 처리가능하며, J2EE, J2SE 환경의 어플리케이션에 사용 가능하다. 수행 할 Job 은 POJO(Plain Old Java Object)로 직접 정의할 수도 있고 Glue 에서 제공하는 Class 를 사용해서 특정 서비스를 실행 시킬 수도 있다.

15.1 Spring Scheduler

Spring 3 에서 부터는 간단하게 비동기 실행과 Task 스케줄링을 TaskScheduler 인터페이스에 대한 추상화를 통해 제공하고 있다.

TaskScheduler 는 Execution 대상이 되는 Task 를 특정 시점 이후에 한 번 실행하거나 fixedRate 또는 fixedDelay 정보를 기반으로 주기적으로 실행할 수 있는 메소드를 제공하고 있다.

- `<task:scheduler/>` : TaskScheduler(`<task:scheduler/>`)에 대한 속성 정의를 위해 task 라는 Namespace 를 제공한다. 또한 이를 이용하면 간편하게 Task Scheduling(`<task:scheduled-task/>`)을 위한 속성을 정의할 수 있게 된다. task Namespace 를 사용하기 위해서는 해당 XML 파일 내의 `<beans>` 정의시 spring-task.xsd 를 선언해 주어야 한다.
- `<task:scheduled-task/>` : `<task:scheduled-task/>`는 기본적으로 'scheduler'라는 속성을 가지고 있는데 이것은 내부에 정의된 Task 를 Scheduling 하기 위한 TaskScheduler Bean 을 정의하기 위한 것이다. `<task:scheduled-task/>`는 하위에 다수의 `<task:scheduled/>`를 포함할 수 있다.
- `<task:scheduled/>` : `<task:scheduled/>`는 'ref'와 'method'는 속성을 갖고 실행주기를 정의한다. 'ref'는 실행 대상이 되는 Bean 을 정의하고, 'method'는 실행 대상 Bean 의 실행 대상 메소드를 정의한다.

다음은 이러한 것들이 정의된 예제이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:task="http://www.springframework.org/schema/task"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/task
    http://www.springframework.org/schema/task/spring-task.xsd">

  <task:scheduled-tasks scheduler="scheduler">
    <task:scheduled ref="glue-task-1" method="doGlueService" fixed-delay="50000"/>
    <task:scheduled ref="glue-task-1" method="doGlueService" fixed-rate="70000"/>
    <task:scheduled ref="glue-task-1" method="doGlueService"
      cron="*/8 * * * * MON-FRI"/>
    <task:scheduled ref="glue-task-2" method="doGlueService" fixed-delay="80000"/>
  </task:scheduled-tasks>
</beans>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

</task:scheduled-tasks>
<task:scheduler id="scheduler" pool-size="3" />

<bean id="glue-task-1"
      class="com.poscoict.glueframework.scheduling.task.GlueTaskScheduler">
  <property name="ServiceName" value="task1-service"/>
</bean>
<bean id="glue-task-2"
      class="com.poscoict.glueframework.scheduling.task.GlueTaskScheduler">
  <property name="ServiceName" value="task2-service"/>
</bean>
</beans>

```

예제의 <task:scheduler/>는 pool-size, id 속성이 사용되었다. 그래서 정의된 Pool Size 를 기반으로 ThreadPoolTaskScheduler 인스턴스가 생성될 것이다. 정의된 id 는 Pool 에 관리될 Task Thread 의 Prefix 로 사용된다.

<task:scheduled/> 의 실행주기는 fixed-delay, fixed-rate, cron 속성을 통해 정의할 수 있다.

- fixed-delay : 이전에 실행된 Task 의 종료 시간으로부터의 fixed-delay 로 정의한 시간만큼 소비한 이후 Task 실행. (Milliseconds 단위로 정의)
- fixed-rate : 이전에 실행된 Task 의 시작 시간으로부터 fixed-rate 로 정의한 시간만큼 소비한 이후 Task 실행. (Milliseconds 단위로 정의)
- cron : Cron Expression 을 이용하여 Task 실행 주기 정의. Cron Expression 은 6 개의 Field 로 구성되며 각 Field 는 순서대로 second, minute, hour, day, month, weekday 를 의미한다. 각 Field 의 구분은 Space 로 한다. 또한 month 와 weekday 는 영어로 된 단어의 처음 3 개의 문자로 정의할 수 있다.

Cron 표현	의미
0 0 * * * *	매일 매시 시작 시점
* / 10 * * * * *	10 초 간격
0 0 8-10 * * *	매일 8,9,10 시
0 0 / 30 8-10 * * *	매일 8:00, 8:30, 9:00, 9:30, 10:00
0 0 9-17 * * MON-FRI	주중 9 시부터 17 시까지
0 0 0 25 12 ?	매년 크리스마스 자정

[그림 67] Cron Expression

org.springframework.scheduling.support.CronSequenceGenerator API 참조

15.1.1 GlueTaskScheduler

<task:scheduled ref=... method=.../> 에 정의되는 Task Class 로 Glue Framework 에서는 특정 서비스를 시행 시킬 수 있는 GlueTaskScheduler 클래스를 제공하고 있다. 그러므로 XML 에 Task Class 로 com.poscoict.glueframework.scheduling.GlueTaskScheduler 등록 하고 <task:scheduled../>에 method 로 doGlueService 를 설정하면 된다.

```

<task:scheduled-tasks scheduler="scheduler">
  <task:scheduled ref="glue-task-1" method="doGlueService" fixed-delay="50000"/>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<task:scheduled ref="glue-task-2" method="doGlueService" fixed-rate="70000"/>
<task:scheduled ref="glue-task-3" method="doGlueService"
    cron="*/8 * * * * MON-FRI"/>
<task:scheduled ref="glue-task-4" method="doGlueService" fixed-delay="80000"/>
</task:scheduled-tasks>
<task:scheduler id="scheduler" pool-size="3" />

```

com.poscoict.glueframework.scheduling.GlueTaskScheduler 는 다음과 같은 3 개 property 를 제공한다.

- ServiceName : 필수. Glue Service 실행을 위한 ServiceName
- datatMap : 선택. 사용자 data 로 GlueService 에서 사용됨
- logger : 선택. Task 실행 history 관리. GlueScheduledTaskLogger 인터페이스 구현체가 된다.

다음은 GlueTaskScheduler 의 전체 property 를 포함한 예제이다.

```

<bean id="task" class="com.poscoict.glueframework.scheduling.GlueTaskScheduler">
    <property name="ServiceName" value="task1-service"/>
    <property name="dataMap">
        <map>
            <entry key="deptno" value="10"/>
            <entry key="find" value="1"/>
        </map>
    </property>
    <property name="logger" ref="dbLogging"/>
</bean>

```

15.1.2 Task 실행 History

GlueTaskScheduler 는 logger 라는 property 는 GlueScheduledTaskLogger 인터페이스 구현체를 필요로 한다. GlueScheduledTaskLogger 의 구현체로 GlueScheduledTaskDBLoggerImpl 을 사용할 수 있으며, dataSource property 를 필요로 한다.

dataSource 의 autoCommit 속성은 true 가 될수 있도록 한다. 그리고 dataSource 는 task 의 실행 History 를 위해서만 사용되도록 한다.

다음은 GlueTaskScheduler 의 logger 참조 bean 의 설정 예이다.

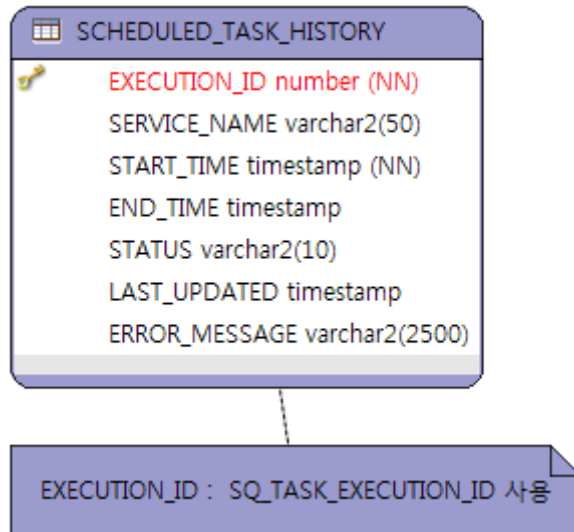
```

<bean id="dbLogging"
    class="com.poscoict.glueframework.scheduling.store.GlueScheduledTaskDBLoggerImpl">
    <property name="dataSource" ref="logging-ds"/>
</bean>
<bean id="logging-ds"
    class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
    <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
    <property name="url" value="jdbc:oracle:thin:@127.0.0.1:1521:XE"/>
    <property name="username" value="scott"/>
    <property name="password" value="tiger"/>
    <property name="defaultAutoCommit" value="true"/>
</bean>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

GlueScheduledTaskDBLoggerImpl 은 SCHEDULED_TASK_HISTORY 라는 table 을 필요로 한다.
glue-schedule 모듈에 oracle 용 table 생성 script 을 포함하고 있다.



[그림 68] ERD

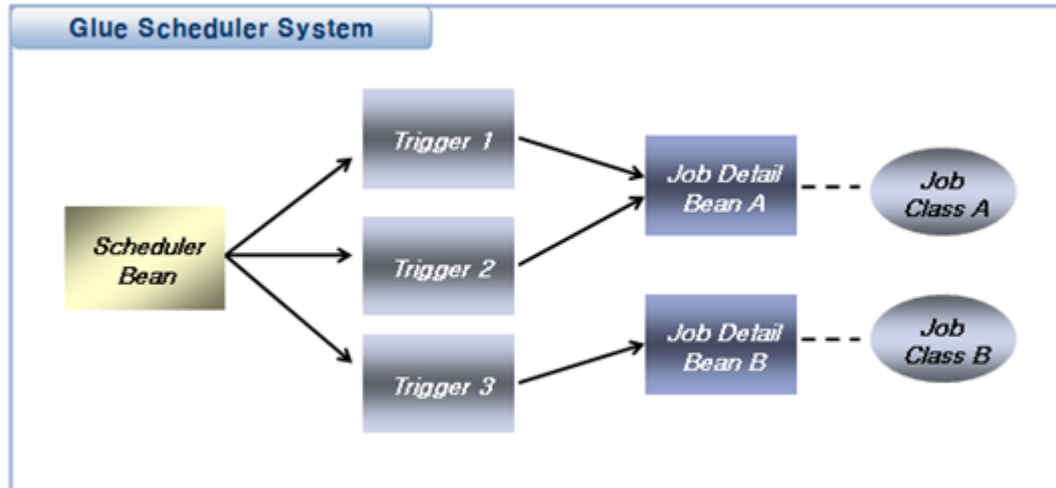
다음은 SCHEDULED_TASK_HISTORY 의 컬럼 설명이다.

- EXECUTION_ID : PRIMARY KEY(자동 증가)
- SERVICE_NAME : 실행된 Glue Service 명
- START_TIME : Task 실행 시작 시간
- END_TIME : Task 실행 종료 시간
- STATUS : 상태(RUNNING:실행 중 | COMPLETED: 완료 | ERROR:에러 발생)
- LAST_UPDATED : db logging time
- ERROR_MESSAGE : 에러 메시지

15.2 Quartz Job Scheduler

Glue Framework 에서는 오픈소스인 Quartz Job Scheduler 를 확장하여 스케줄러 기능을 제공한다.
TaskScheduler 기반의 Scheduler 보다 설정 방법은 복잡하지만 TaskScheduler 로 처리가 어려운
복잡한 스케줄링을 처리할 수 있다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0



[그림 69] Glue Scheduler Conceptual Diagram

하나의 Scheduler Bean 에 다수의 Trigger 를 등록할 수 있다. Trigger 에는 Job 의 정주기 정보를 정의한다. (예, 매일 오후 7 시에 실행한다.) 자세한 정의 방법은 다음 섹션의 개발 예제를 참고하기 바란다.

Diagram 에서 보면 알 수 있듯이 Job Detail 은 복수 개의 Trigger 를 기반으로 작동될 수 있다. Job Class 역시 다수의 Job Detail 에서 재사용될 수 있다. 즉, 비즈니스 로직을 구현한 Job Class 는 다양한 정주기 패턴에 따라 계속적으로 적용이 가능하다.

(예를 들면, 같은 업무를 수행하는 Job Class 를 매일 18 시에 수행하도록 하고, 또 매 월 15 일에는 9 시에 한 번 더 실행이 되도록 구성을 할 수 있다는 것이다.)

실제 비즈니스 로직을 구현하는 Job Class 는 GlueQuartzJobBean 이며 executeJob 메소드를 구현해야 한다. Job Class 이외의 Scheduler Bean, Trigger, JobDetail 정보는 모두 xml 설정 파일에 정의한다.

Conceptual Diagram 을 xml 로 다음과 같이 표현할 수 있다. 개발자가 실질적으로 코딩하는 부분은 Job Class 하나이지만, 이 클래스가 정상적으로 작동하기 위해서는 xml 에 Job 의 정주기 정보를 등록해야 한다. (Trigger 정보)

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

  <bean id="scheduler"
    class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
    <property name="triggers">
      <list>
        <ref local="simpleTrigger1"/>
        <ref local="simpleTrigger2"/>
        <ref local="cronTrigger"/>
      </list>
    </property>
  </bean>
  <bean id="simpleTrigger1"
    class="org.springframework.scheduling.quartz.SimpleTriggerBean">
    <property name="jobDetail" ref="jobDetail-A"/>
    <property name="repeatInterval" value="10000"/>
  </bean>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<bean id="simpleTrigger2"
      class="org.springframework.scheduling.quartz.SimpleTriggerBean">
    <property name="jobDetail" ref="jobDetail-B"/>
    <property name="repeatInterval" value="10000"/>
</bean>
<bean id="cronTrigger"
      class="org.springframework.scheduling.quartz.CronTriggerBean">
    <property name="jobDetail" ref="jobDetail-A"/>
    <property name="cronExpression" value="0 40 18 * * ?"/>
</bean>
<bean id="jobDetail-A"
      class="org.springframework.scheduling.quartz.JobDetailBean">
    <property name="jobClass"
              value="com.poscoict.glueframework.scheduling.GlueQuartzJobBean"/>
    <property name="jobDataAsMap">
        <map>
            <entry key="ServiceName" value="job1-service"/>
        </map>
    </property>
</bean>
<bean id="jobDetail-B"
      class="org.springframework.scheduling.quartz.JobDetailBean">
    <property name="jobClass"
              value="com.poscoict.glueframework.scheduling.GlueQuartzJobBean"/>
    <property name="jobDataAsMap">
        <map>
            <entry key="ServiceName" value="job2-service"/>
        </map>
    </property>
</bean>
</beans>

```

Scheduler, Trigger, Jobdetail 을 좀더 자세히 살펴보자.

Scheduler 를 정의 하기 위해서는 com.poscoict.glueframework.scheduling.GlueJobScheduler 를 사용하여야 하며, GlueJobScheduler 는 다음과 같은 property 를 갖는다.

프로퍼티	설명
startupDelay	스케줄러오브젝트가생성된후서비스를시작할때까지대기하는시간. 보통 10~20 초로설정하는것을권장한다. (다른애플리케이션의 standby 상태를고려) 디폴트값은 0. (단위 second)
triggers	Job Detail 을수행시켜줄 Trigger 를리스트형태로등록한다. Trigger 의 Bean ID 를기재한다.
jobDetails	
globalJobListeners	

[표 18] GlueJobScheduler 의 property

Trigger 는 job 의 실행주기를 정의한다. Trigger 는 job 을 실행시켜주는 방아쇠(Trigger) 역할을 한다. Trigger 는 SimpleTrigger, CronTrigger 두 종류가 제공되며 다음과 같은 property 를 갖는다.

property	설명	대상
----------	----	----

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

jobDetail	Trigger 가동할 JobDetail 의 ID 를기재한다.	SimpleTriger, CronTrigger
startDelay	Job 이기동된후대기시간을설정한다. 보통지정하지않는다.	SimpleTriger, CronTrigger
repeatCount	Job 수행횟수를제한한다. 디폴트값은무한대이다.	SimpleTriger, CronTrigger
repeatInterval	Job 수행간격을정의한다. (단위; milliseconds)	SimpleTriger, CronTrigger
cronExpression	Cron 표현식을정의한다. * 모든값 ? 특정값없음 - 범위지정에사용 , 여러값지정구분에사용 / 초기값과증가치설정에사용 - L 지정할수있는범위의마지막값 - W 월~금요일또는가장가까운월/금요일 # 몇번째무슨요일 2#1 =>첫번째월요일 예제) Expression Meaning 초분시일월주(년) "0 0 12 * * ?" 아무요일, 매월, 매일 12:00:00 "0 15 10 ? * *" 모든요일, 매월, 아무날이나 10:15:00 "0 15 10 * * ?" 아무요일, 매월, 매일 10:15:00 "0 15 10 * * ? *" 모든연도, 아무요일, 매월, 매일 10:15 "0 15 10 * * ? 2005" 2005 년아무요일이나매월, 매일 10:15 "0 * 14 * * ?" 아무요일, 매월, 매일, 14 시매분 0 초 "0 0/5 14 * * ?" 아무요일, 매월, 매일, 14 시매 5 분마다 0 초 "0 0/5 14,18 * * ?" 아무요일, 매월, 매일, 14 시, 18 시매 5 분마다 0 초 "0 0-5 14 * * ?" 아무요일, 매월, 매일, 14:00 부터매 14:05 까지매분 0 초 "0 10,44 14 ? 3 WED" 3 월의매주수요일, 아무날짜나 14:10:00, 14:44:00 "0 15 10 ? * MON-FRI" 월~금, 매월, 아무날이나 10:15:00 "0 15 10 15 * ?" 아무요일, 매월 15 일 10:15:00 "0 15 10 L * ?" 아무요일, 매월마지막날 10:15:00 "0 15 10 ? * 6L" 매월마지막금요일아무날이나 10:15:00 "0 15 10 ? * 6L 2002-2005" 2002 년부터 2005 년까지매월마지막금요일아무날이나 10:15:00 "0 15 10 ? * 6#3" 매월 3 번째금요일아무날이나 10:15:00	CronTrigger

[표 19] Trigger 의 Property

JobDetail 은 실제 비즈니스로직을구현한 Job Class 정보와 Job 수행에 필요한 파라미터 정보를 관리한다.

프로퍼티	설명
jobClass	Full package 로기재한다.
jobDataAsMap	Job 수행에필요한파라미터를 map 형태로등록한다.
description	Job 설명을기재한다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

[표 20] JobDetail 의 Property

15.2.1 GlueQuartzJobBean

JobDetail 의 참조 bean 은 다음과 같은 형태이다.

```
<bean id="JobB" class="org.springframework.scheduling.quartz.JobDetailBean">
  <property name="jobClass"
    value="com.poscoict.glueframework.scheduling.GlueQuartzJobBean"/>
  <property name="description" value=" service 파일이용"/>
  <property name="jobDataAsMap">
    <map>
      <entry key="ServiceName" value="JobB-service"/>
      <entry key="job" value="manager"/>
    </map>
  </property>
</bean>
```

GlueQuartzJobBean 을 이용하는 경우는 특정 Glue service 를 실행 시킬 수 있다. jobDataAsMap 의 ServiceName 에 실행 시킬 Glue service 명을 정의 하면 된다.

Glue Sevice 실행 전과 후에 추가적인 로직을 넣고자 한다면, GlueQuartzJobBean 을 상속해서 구현할 수 있다. 다음 2 개 methde 에서 service 수행 전후 필요한 일을 구현 한다.

```
public class JobD extends GlueQuartzJobBean
{
  protected void beforeExecuteJob(JobExecutionContext context)
  {
    //service 수행전 사전 작업이 있다면 구현한다.
    //JobDetail 의 jobDataAsMap 에 담지 못한 Data 를 추가로 담는다.
    context.getJobDetail().getJobDataMap().put("name", value);
  }
  protected void afterExecuteJob(JobExecutionContext context)
  {
    //service 에 대한 수행이 완료된 후 필요한 작업이 있다면
    //구현한다.
  }
}
```

15.2.2 Job 실행 History

Job 실행 history 는 scheduler bean 의 globalJobListeners property 를 이용한다.

```
<bean id="scheduler"
  class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
  <property name="triggers" . . ./>
  <property name="globalJobListeners">
    <list>
      <ref local="dblogging"/>
    </list>
  </property>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

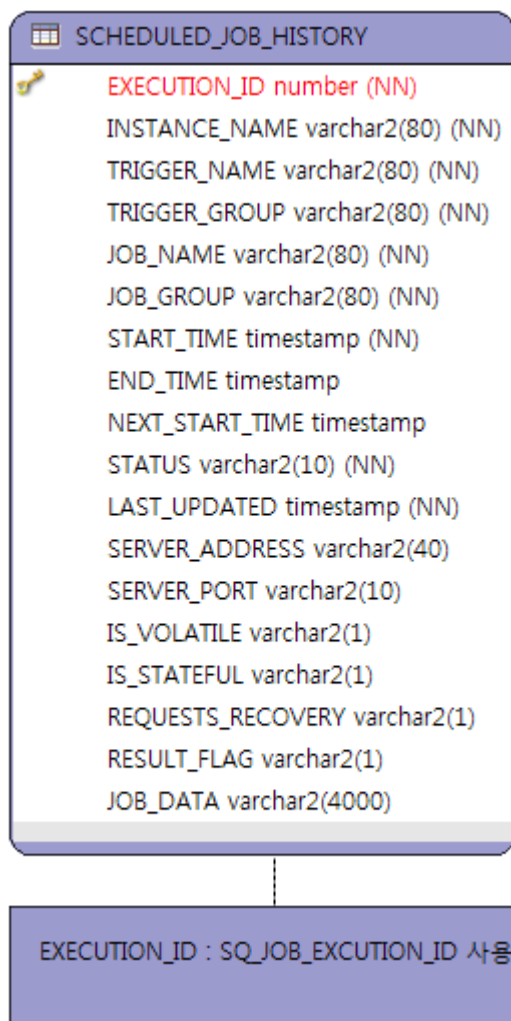
```

</bean>
<bean id="dblogging"
      class="com.poscoict.glueframework.scheduling.store.GlueScheduledJobInfoLogger">
  <property name="dataSource" ref="logging-ds"/>
</bean>
<bean id="logging-ds"
      class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="url" value="jdbc:oracle:thin:@127.0.0.1:1521:XE"/>
  <property name="username" value="scott"/>
  <property name="password" value="tiger"/>
  <property name="defaultAutoCommit" value="true"/>
</bean>

```

GlueScheduledJobInfoLogger 를 이용해 Job 수행실적정보를 Database 에 저장할 수 있다. GlueScheduledJobInfoLogger 는 SCHEDULED_JOB_HISTORY 라는 table 을 필요로 한다. glue-schedule 모듈에 oracle 용 table 생성 script 을 포함하고 있다.

dataSource 의 autoCommit 속성은 true 가 될수 있도록 한다. 그리고 dataSource 는 job 의 실행 History 를 위해서만 사용되도록 한다.



[그림 70] ERD

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

15.3 Scheduler Luanch

Scheduler 의 종류에 따라 편의상 spring-scheduler.xml , quartz-scheduler 로 구분한다.

스케줄러를 기동하는 방법은 크게 세 가지가 있다.

1. 테스트를 목적으로 Standalone JVM 에서 기동하는 경우
2. 서블릿 컨테이너에서 기동하는 경우
3. Glue Http Server 에서 기동하는 경우, 즉 서블릿 컨테이너 외부에서 Scheduler 를 실행하는 경우
테스트 목적이면 1)번 방법을 사용하고, 그렇지 않다면 2)번, 3)번 중에 선택하여 스케줄러를 기동한다.
보통 CPU, Memory, Application Performance 문제의 우려가 있기 때문에 3)번 방법을 권장한다.

15.3.1 Standalone JVM

Main 메소드를 가진 클래스에서 Scheduler 오브젝트를 생성하면 스케줄러가 자동으로 기동된다.

테스트하는 경우에만 사용할 것을 권장한다

Spring Scheduler 일 경우는 다음과 같다.

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Run_SpringScheduler
{
    public static void main( String[] args )
    {
        ApplicationContext context
            = new ClassPathXmlApplicationContext( "spring_scheduler.xml" );
    }
}
```

Query Job Scheduler 의 경우는 다음과 같다.

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Run_SpringScheduler
{
    public static void main( String[] args )
    {
        ApplicationContext context
            = new ClassPathXmlApplicationContext( "quartz_scheduler.xml" );
    }
}
```

15.3.2 서블릿 컨테이너

스케줄러를 위한 별도의 물리적인 서버가 가용하지 않을 때 사용하는 방법이다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

관련 Bean 을 인스턴스화하는 Servlet Context Listener 를 web.xml 에 등록한다. 서블릿 컨테이너가 시작하면서 스케줄러가 자동적으로 Start 될 것이다.

다음은 spring scheduler 를 사용할 경우이다.

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring_scheduler.xml</param-value>
</context-param>
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

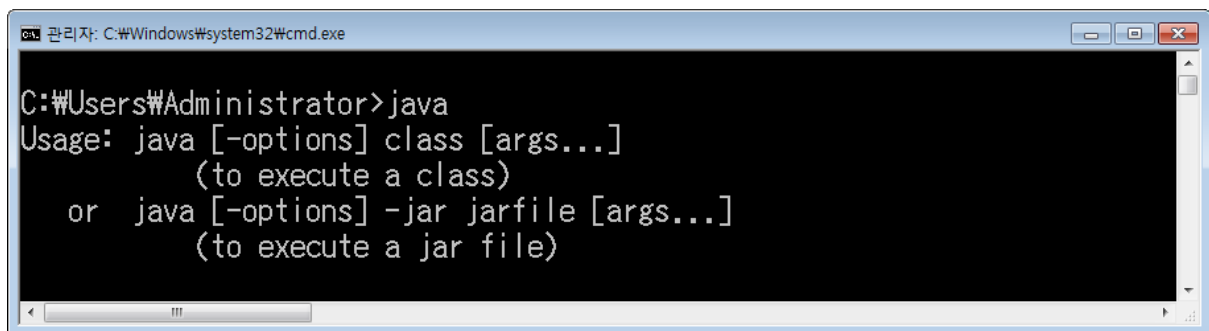
다음은 quartz job scheduler 를 사용할 경우이다.

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/quartz_scheduler.xml</param-value>
</context-param>
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

15.3.3 Glue Scheduler Server

mina 기반의 스케줄러만을 위한 별도의 Server 이다. 서버가 기동되면, Quartz Job Scheduler 에 한해 remote 에서 요청되는 Job 실행 이벤트까지 처리가능하다.

java 사용법은 다음과 같다.



[그림 71] java Usage

그리고 Glue 에서는 Scheduler 실행시 java -jar jarfile 을 사용할 것이다. 실행시 args 에 따라 scheduler 를 지정할 수 있다.

scheduler 관련 options 에는 다음과 같은 것이 있다. 그외 java option 을 추가할 수 있다.

- glue.scheduler.server.port : scheduler server 에서 사용할 port. 8805 를 default 로 사용함.
- glue.scheduler.server.address : scheduler server 에 할당된 ip

args 는 2 개를 필요로 하며 다음과 같다.

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

- scheduler type : spring 또는 quartz. Default 는 'quartz' 임
- scheduler xml file 명 : xml file 명. 'Default 는 quartz_scheduler.xml' 임

다음은 spring scheduler 일 경우의 실행방법이다.

```
java -jar application.jar spring spring_scheduler.xml
```

다음은 quartz job scheduler 일 경우의 실행방법이다.

```
java -jar application.jar spring quartz_scheduler.xml
```

다음은 java option 이 사용된 실행방법이다.

```
java -Dglue.scheduler.server.port=8805 -DCONFIG_PATH=C:/ application.jar spring quartz_scheduler.xml
```

15.3.3.1 Ant 설정

build.properties 를 다음과 같다. 빌드를 위한 기본 정보를 다음과 같이 설정한다.

```
JAR_NAME=application
build.dir=./build
class.dir=./classes
lib.dir=./build/lib
GlueSDK.dir=C://eclipse/users/GlueSDK
```

build.xml 은 다음과 같다. jar 파일 생성시 MENIFEST.MF 파일에 Main-Class 와 ClassPath 가 포함되도록 작성한다.

```
<project name="GlueSample" default="user-application">
  <property file="build.properties"/>
  <target name="makeBuilDir">
    <mkdir dir="${build.dir}"/>
    <mkdir dir="${build.dir}/lib"/>
  </target>
  <path id="class.path">
    <fileset dir="${GlueSDK.dir}/lib/gluelib" >
      <include name="*.jar"/>
    </fileset>
    <fileset dir="${GlueSDK.dir}/lib/gluestd" >
      <include name="*.jar"/>
    </fileset>
  </path>
  <pathconvert property="class-path" pathsep=" " dirsep="\">>
    <path refid="class.path"></path>
    <map from="${GlueSDK.dir}/lib/gluestd" to="lib"/>
    <map from="${GlueSDK.dir}/lib/gluelib" to="lib"/>
  </pathconvert>
  <target name="create-manifest" depends="">
    <tstamp>
```

```

        <format property="TODAY_KO"
            pattern="yyyyMMdd.HH:mm"
            timezone="GMT+9" locale="ko"/>
    </tstamp>
    <manifest file="${build.dir}/MANIFEST.MF">
        <attribute name="Main-Class"
            value="com.poscoict.glueframework.scheduling.server.GlueSchedulerHttpServer"/>
        <attribute name="Class-Path" value="${class-path}" />
    </manifest>
</target>
<target name="user-application" depends="makeBuildDir,create-manifest">
    <jar destfile="${build.dir}/${JAR_NAME}.jar"
        manifest="${build.dir}/MANIFEST.MF" compress="false">
        <fileset dir="${class.dir}">
            <patternset>
                <include name="**/*.*/>
            </patternset>
        </fileset>
    </jar>
    <copy todir="${build.dir}/lib">
        <fileset dir="${GlueSDK.dir}/lib/gluelib">
            <include name="*.jar"/>
        </fileset>
        <fileset dir="${GlueSDK.dir}/lib/gluestd">
            <include name="*.jar"/>
        </fileset>
    </copy>
</target>
</project>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

16. Spring Mobile

16.1 spring mobile 라이브러리설정

Glue에서는기본적으로 1.1.0.M3 버전의 Spring Mobile(예:spring-mobile-device-1.1.0.M3.jar)를사용한다.(<http://www.springsource.org/spring-mobile> 참조)

1.1.0.M3 이전버전의라이브러리를사용할경우 Glue에서사용하는 Spring 3.2.0 버전이아니라그이전버전과호환되면서동작하지않는기능들이있으므로주의해야한다.

그런데 Maven 을사용할경우 Maven 에서제공하는최신버전((2013 년 5 월 16 일기준))은 1.0.1.RELEASE 인데이버전은 Spring 3.1.1 과호환되므로 1.1.0.M3 이상버전의라이브러리를프로젝트에서자체적으로사용하는 Maven Repository 서버에올려놓거나공용으로사용하는서버가없다면로컬의 Maven Repository 에올려놓아야한다.

예를들어로컬의 Maven Repository 가'E:\mvn\MavenRepository'로정의되어있다면 spring-mobile-device-1.1.0.M3.jar 파일을직접'E:\mvn\MavenRepository\org\springframework\mobile\spring-mobile-device\1.1.0.M3'폴더에배치해주어야한다.

pom.xml 에는아래와같이 dependency 를추가해주면된다.

```
<dependency>
  <groupId>org.springframework.mobile</groupId>
  <artifactId>spring-mobile-device</artifactId>
  <version>1.1.0.M3</version>
</dependency>
```

16.2 interceptor

Spring Mobile 에서는 DeviceResolverHandlerInterceptor, SiteSwitcherHandlerInterceptor 를 제공한다.

다음은 DeviceResolverHandlerInterceptor 와 SiteSwitcherHandlerInterceptor 를 xxx-servlet.xml 에 설정한 예이다.

```
<bean id="urlMapping"
  class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <value>
      /sample1.mvc=controller
    </value>
  </property>
  <property name="interceptors">
    <list>
      <ref bean="deviceResolverHandlerInterceptor"/>
      <ref bean="siteSwitcherHandlerInterceptor"/>
    </list>
  </property>
</bean>
<bean name="deviceResolverHandlerInterceptor"
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

class="org.springframework.mobile.device.DeviceResolverHandlerInterceptor" />
<bean name="siteSwitcherHandlerInterceptor"
class="org.springframework.mobile.device.switcher.SiteSwitcherHandlerInterceptor"
factory-method="mDot" >
<constructor-arg value="test.com" />
</bean>

```

16.2.1 DeviceResolverHandlerInterceptor

DeviceResolverHandlerInterceptor 는 Request 의 header 를 분석하여 어떤 Device 인지 관련정보를 Device 객체로 만들어서 Request 에 담아 주는 역할을 한다.

DeviceResolverHandlerInterceptor 에 의해 Request 에 Device 객체가 담기며, Device 객체의 접근을 DeviceUtils 의 getCurrentDevice 메서드를 사용하면 된다.

JSP, Controller, Interceptor 와 같은 부분에서는 request 를 사용할 수 있으므로 다음과 같이 사용한다.

```
Device device = DeviceUtils.getCurrentDevice(request);
```

Glue Service 에서는, 즉 Activity 에서는 GlueContext 를 통해 다음과 같이 Device 객체를 가져온다.

```
Device device = (Device)ctx.get(DeviceUtils.CURRENT_DEVICE_ATTRIBUTE);
```

이러한 Device 객체를 이용하면 Client 가 일반 PC 의 브라우저인지, 태블릿인지, Mobile 기기인지 구분 할 수 있다.

```

if(device.isMobile()){
//Mobile 기기인경우
}else if(device.isTablet()){
//Tablet 기기인경우
}else if(device.isNormal()){
//일반 PC 인경우
}

```

다음은 추가적인 Interceptor 를 통해 Device 객체를 사용하여 Mobile 페이지를 별도로 설정하는 예이다. 일반 PC 에서 접근할 경우와 Mobile 을 통해서 접근할 때 비즈니스 로직은 동일하고 뷰페이지만 다르다면 다음과 같이 Intercetor 를 작성하여 Mobile 기기로 접근할 경우 일반 뷰페이지가 아닌 Mobile 용 페이지를 보여줄 수 있다.

```

public class MobileSampleInterceptor extends HandlerInterceptorAdapter
{
/** logger */
private GlueLog logger = GlueLogFactory.getLogger(getClass());

@Override
public void postHandle(HttpServletRequest request, HttpServletResponse response,
Object handler, ModelAndView modelAndView)
throws Exception
{
Device device = DeviceUtils.getCurrentDevice(request);
String viewName = modelAndView.getViewName();
if(device.isMobile()){
viewName = viewName+"_mobile";
}
}
}

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

        modelAndView.setViewName(viewName);
    }
    logger.info("viewName=={}",viewName);
}
}

```

위와 같은 Intercetor 가 설정되었고, PC에서는 sample.jsp를 뷰페이지로 사용한다면, Mobile 기기에서 접근하는 경우에는 sample_mobile.jsp를 뷰페이지로 사용하게 된다.

16.2.2 SiteSwitcherHandlerInterceptor

SiteSwitcherHandlerInterceptor는 Mobile 기기로 접근하는 Client를 다른 URL로 Redirect시키는 역할을 한다. SiteSwitcherHandlerInterceptor를 Interceptor로 설정할 경우에는 반드시 DeviceResolverHandlerInterceptor과 같이 설정하여야 한다.

SiteSwitcherHandlerInterceptor의 factory-method로 mDot, dotMobi, urlPath를 설정할 수 있다.

```

<bean name="case1"
      class="org.springframework.mobile.device.switcher.SiteSwitcherHandlerInterceptor"
      factory-method="mDot" >
    <constructor-arg value="test.com" />
</bean>

<bean name="case2"
      class="org.springframework.mobile.device.switcher.SiteSwitcherHandlerInterceptor"
      factory-method="dotMobi" >
    <constructor-arg value="test.com" />
</bean>

<bean name="case3"
      class="org.springframework.mobile.device.switcher.SiteSwitcherHandlerInterceptor"
      factory-method="urlPath" >
    <constructor-arg value="/m" />
</bean>

```

접근하고자 하는 url이 http://www.test.com:8080/GlueSample/sample.mvc라면 factory-method별로 constuctor-arg의 값을 이용해 다음과 같이 redirect된다.

factory-method	Redirect url
mDot	http://m.test.com:8080/GlueSample/sample.mvc http://www.test.com:8080 → http://m.test.com:8080 으로 변경되었음.
dotMobi	http://www.test.mobi:8080/GlueSample/sample.mvc http://www.test.com:8080 → http://www.test.mobi:8080 으로 변경되었음.
urlPath	http://www.test.com:8080/m/GlueSample/sample.mvc http://www.test.com:8080/ → http://www.test.com:8080/m/ 으로 변경되었음.

[표 21] SiteSwitcherHandlerInterceptor의 factory-method

16.3 WURFL

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

WURFL 란

1. Wireless Universal Resource File 의약자로오픈데이터모바일장비데이터베이스와오픈소스 API 를제공한다.
2. 커뮤니티주도하에모바일장비정보데이터베이스를유지한다.
3. XML 형식으로장비정보데이터를위해대용량 XML 파일을사용하고있다.
4. API 는특성과그룹명으로검색가능하다.
5. 데이터베이스는계층구조를이루고있다.
6. HTTP 요청또는 User-Agent 헤더값으로모바일장비를인식할수있는메소드제공한다.
7. 그룹이나, 특성이름으로장비정보를얻는메소드제공한다.

기타 자세한 정보는 WURFL 사이트(<http://wurfl.sourceforge.net>)를 참고하세요.

wurfl 을 사용하기 위해서는 java 라이브러리와 장비 정보가 담긴 xml 파일이 필요하며, 두 파일 모두 WURFL 사이트에서 다운로드 받을 수 있다.

Java 라이브러리는 기본적으로 1.4 버전(wurfl-1.4.jar)을 사용하며 Maven 을 사용할 경우에는 Maven 에서 제공하는 최신버전(2013 년 5 월 16 일기준)은 1.3.1.1 이므로 WURFL 사이트에서 다운받아 Maven Repository 에 배치해야 한다.

장비 정보가 담긴 xml 파일은 zip 으로 제공되며 현재(2013 년 5 월 16 일기준)버전은 2.3.3(wurfl-2.3.3.zip)이다. 장비 정보는 계속 업데이트되므로 가능하면 사이트를 통해서 최신버전을 확인하여 다운받도록 한다.

Glue 기반에서 WURFL Java API 를 사용하기 위해서는 applicationContext.xml 에 net.sourceforge.wurfl.core.GeneralWURFLEngine 클래스를 Bean 으로 등록하여 관련 기능들을 사용할 수 있다. constructor-arg 에는 장비 정보가 담긴 zip 파일의 Path 를 설정한다.

```
<bean id="wurflHolder" class="net.sourceforge.wurfl.core.GeneralWURFLEngine">
    <constructor-arg index="0" value="classpath:/wurfl-2.3.3.zip" />
</bean>
```

참고로 1.0.0.M4 버전까지는 Spring Mobile 의 DeviceResolverHandlerInterceptor 와 연동하여 설정하는 기능을 제공했다.

GeneralWURFLEngine 을 통해서 WURFLManager 를 얻어올 수 있고 WURFLManager 는 Request 정보를 기반으로 XML 의 장비 정보를 가진 Device 객체를 가져올 수 있다.

```
GeneralWURFLEngine wurfl
=
(GeneralWURFLEngine)GlueStaticContext.getBeanFactory().getBeanObject("wurflHolder");
WURFLManager manager = wurfl.getWURFLManager();
net.sourceforge.wurfl.core.Device device = manager.getDeviceForRequest(request);

logger.info("WURFL Device : {}",device);
logger.info("getCapabilities : {}",device.getCapabilities());
logger.info("getMatchType : {}",device.getMatchType());
logger.info("getMarkUp : {}",device.getMarkUp());
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

WURFL 의 Device(net.sourceforge.wurfl.core.Device)는 Spring Mobile 의 Device (org.springframework.mobile.device.Device) 와는 다르므로 혼동하지 않도록 주의해야 한다.

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

17. AUDIT

17.1 Audit 기능 개요

Glue에서는 DB Table 들에 Insert 혹은 Update 를 수행할 때 공통으로 Update 하는 컬럼이 있을 경우 GlueContext 설정된 auditAttributes 값을 자동으로 Binding 해 주는 기능을 제공하고 있다.

예를 들어 모든 Table 들이 UPDATE_DT 라는 컬럼을 가지며 Insert 혹은 Update 를 수행할 때 해당 컬럼에 항상 현재 시간을 Update 할 경우 GlueContext 의 auditAttributes 에 해당 값을 설정하고 GlueDao 클래스를 사용하여 Insert 혹은 Update 를 수행할 때 UPDATE_DT 에 자동으로 현재 시간을 Binding 해 줄 수 있다.

Glue 에서 제공하는 Audit 기능은 기본적으로 아래와 같은 흐름으로 동작한다.

1. GlueContext 를 생성하여 Glue Service 를 실행하기 전에 glue.properties 에 Audit 항목이 설정되어 있는지를 확인하고 설정되어 있다면 GlueAuditAttributes 를 생성하여 GlueContext 에 설정한다
2. GlueDao 클래스를 사용하여 Insert 혹은 Update 를 수행하기전 GlueParameter 를 생성할때 GlueContext 에 auditAttributes 가 설정되어 있다면 GlueParameter 에 해당 값을 설정한다.
3. GlueDao 클래스를 사용하여 Insert 혹은 Update 를 수행할 때 GlueParameter 에 auditAttributes 가 설정되어 있다면 해당 항목에 auditAttributes 의 값을 Binding 하여 SQL 실행

17.2 Audit 설정

17.2.1 Audit 항목 설정(glue.properties)

Audit 기능을 사용하기 위해서는 glue.properties 에 Audit 항목을 정의해야 한다.

Insert 항목은 'audit.insert.key' Update 항목은 'audit.update.key' 를 Key 로 정의한다.

여러개의 Audit 항목이 존재 할 경우 ,(컴마)를 구분자로 사용하며 해당 항목에 현재 시간을 timestamp 로 Update 할 경우에는 #current_timestamp# 를 붙여주고 Date 로 Update 할 경우 #current_date# 를 붙여준다.

```
audit.insert.key=cid,cip,cdt#current_timestamp#,uid,uip,udt#current_timestamp#
audit.update.key=uid,uip,udt#current_timestamp#
```

17.2.2 GlueContext 에 GlueAuditAttributes 설정

아래 Class 를 통해서 Glue 서비스가 실행되는 경우에는 서비스 실행 전에 glue.properties 에 Audit 항목이 설정되어 있는지를 확인하고 설정되어 있다면 GlueAuditAttributes 를 생성하여 GlueContext 에 설정하는 로직이 포함되어 있다.

```
com.poscoict.glueframework.web.control.struts.GlueSimpleAction
com.poscoict.glueframework.web.control.spring.GlueSimpleController
com.poscoict.glueframework.web.control.spring.GlueXMLViewController
com.poscoict.glueframework.web.control.restful.RestfulController
com.poscoict.glueframework.web.control.portlet.GlueSimpleController
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
com.poscoict.glueframework.scheduling.GlueQuartzJobBean
com.poscoict.glueframework.scheduling.task.GlueTaskScheduler
```

위 Class 이외의 Class 기반에서 Glue Service 를 실행할 경우에는 GlueContext 에 GlueAuditAttributes 설정하는 기능을 별도로 추가해야 해당 Service 에서 Audit 기능이 정상적으로 작동할 수 있다.

```
if ( GlueStaticContext.isAudit() ) //glue.properties 에 Audit 항목이 설정되어 있는지 판단
{
    //GlueContext 의 값을 기반으로 GlueAuditAttributes 를 생성하여 설정
    //GlueContext 에 없는 값을 기반으로 할 경우에는 Map 기반으로 GlueAuditAttributes 를
    생성(new GlueAuditAttributes( map ))하는 것도 가능
    ctx.setAuditAttributes( new GlueAuditAttributes( ctx ) );
}
```

17.2.3 GlueParameter 에 auditAttributes 설정

GlueDao 클래스를 사용하여 Insert 혹은 Update 를 수행할 때 Audit 기능이 정상적으로 동작하기 위해서는 GlueParameter 에 auditAttributes 가 설정 되어 있어야 한다.

GlueContext 에 auditAttributes 가 설정되어 있다면 해당 값을 가져와서 설정하면 되고 그렇지 않은 경우에는 직접 GlueAuditAttributes 를 생성하여 설정해야 한다.

```
GlueAuditAttributes audit = ctx.getAuditAttributes();
HashMap<String, Object> paramMap = new HashMap<String, Object>();
paramMap.put( "sal", 7000 );
paramMap.put( "ename", "USER8" );
paramMap.put( "empno", 1118 );
paramMap.put( "deptno", 10 );

param = new GlueParameter<HashMap<String, Object>>( paramMap );
param.setAuditAttributes( audit );
dao.insert( "emp.mapper.insertEmpAudit", param );
```

17.2.4 SQL 작성

현재 Audit 기능은 GlueJdbcDao 와 GlueMybatisDao 를 사용할 경우에만 가능하다.

GlueJdbcDao 에서 기본 SQL 을 사용할 경우에는 Audit 항목을 맨 앞쪽에 위치해야 한다.

```
insert into
EMP_AUDIT(CREATE_ID,CREATE_IP,CREATE_DT,UPDATE_ID,UPDATE_IP,UPDATE_DT,EMPNO, ENAME,
SAL, DEPTNO)
values(?,?,?,?,?,?,?,?,?,?)
```

GlueJdbcDao 에서 Named SQL 을 사용할 경우에는 위치와 상관없이 Audit 항목이 있으면 된다.

```
insert into EMP_AUDIT
(EMPNO, ENAME, SAL, DEPTNO, CREATE_ID, CREATE_IP, CREATE_DT, UPDATE_ID, UPDATE_IP,
UPDATE_DT, HIREDATE)
values
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```
(:empno, :ename, :sal, :deptno, :cid, :cip, :cdt, :uid, :uip, :udt, :hiredate)
```

GlueMybatisDao 에서 사용할 경우에는 위치와 상관없이 Audit 항목이 있으면 된다.

```
insert into EMP_AUDIT(EMPNO, ENAME, SAL,
DEPTNO, CREATE_ID, CREATE_IP, CREATE_DT, UPDATE_ID, UPDATE_IP, UPDATE_DT, HIREDATE)
values(#{empno}, #{ename}, #{sal}, #{deptno}, #{create_id}, #{create_ip},
#{create_dt}, #{update_id}, #{update_ip}, #{update_dt}, #{hiredate})
```

단, GlueMybatisDao 를 사용할 경우 다음과 같이 glue.properties 의 값은 DB Column 과 일치 하여야 한다.

```
audit.insert.key=create_id,create_ip,create_dt#current_timestamp#,
update_id,update_ip,update_dt#current_timestamp#
audit.update.key=update_id,update_ip,update_dt#current_timestamp#
```

posco ict

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

18. 기타

이번 장에서는 Glue Framework 기반의 application 개발시 알아야 할 주요 파일들을 다룬다.

18.1 glue-config.xml

glue-config.xml 은 Glue Plug-In 과 관련된 설정 정보를 관리한다. GlueSDK 에 포함되어 있으며, Eclipse IDE 의 Preference 에서 SDK Location 이 지정되면, 그 위치에서 config/glue-config.xml 파일의 설정을 이용하는 것이다.

glue-config.xml 은 activities 와 common 으로 구성된다. activities 에서는 n 개의 activity 로 구성된다. common 은 document, service, query, class-generator, activity-template 로 구성된다.

glue-config.xml 의 XML Schema 정의인 GluePluginConfig.xsd 파일은 glue-schema 모듈에 포함되어 있다. glue-schema 모듈은 GlueSDK/lib/gluelib/glue-schema-{version}.jar 파일이며, jar 파일내 schemaorg_apache_xmlbeans/src 위치에 GluePluginConfig.xsd 가 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<glue-eclipse-config
  xmlns="http://www.poscoict.com/glueframework/plugin/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.poscoict.com/glueframework/plugin/config
    GluePluginConfig.xsd ">

  <activities>
    <activity name="Router" icon="icons/activity_router.gif" addSeperator="true">
      <bubble-tip>Default Router Activity</bubble-tip>
      <template>Glue Router template</template>
      <color>green</color>
      <default-properties>
        <property name="class"
          value="com.poscoict.glueframework.biz.activity.GlueDefaultRouter" />
      </default-properties>
    </activity>
    <activity name="SubService" icon="icons/activity_subservice.gif">
      <bubble-tip>SubService Activity</bubble-tip>
      <template>Glue SubService template</template>
      <color>lightBlue</color>
      <default-properties>
        <property name="class"
          value="com.poscoict.glueframework.biz.activity.GlueSubService" />
        <property name="service-name" value="sample-service" />
        <property name="new-transaction" value="false" />
      </default-properties>
    </activity>
    <!-- 생략 -->
  </activities>
  <common>
    <document>
      <file-type>EUC-KR</file-type>
      <project-name>project-name</project-name>
      <phase>phase</phase>
      <task-name>task-name</task-name>
      <service-id>service-id</service-id>
    </document>
  </common>
</glue-eclipse-config>
```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

        <current-date>current-date</current-date>
        <service-brief>service-brief</service-brief>
        <activity-subject>activity-subject</activity-subject>
        <activity-description>activity-description</activity-description>
    </document>
    <service>
        <file-type>UTF-8</file-type>
    </service>
    <query>
        <file-type>UTF-8</file-type>
    </query>
    <class-generator>
        <exclude-package>com.poscoict.glueframework</exclude-package>
    </class-generator>
    <activity-template>
        <![CDATA[
package #PACKAGE#;

import com.poscoict.glueframework.biz.activity.GlueActivity;
import com.poscoict.glueframework.biz.control.GlueBizControlConstants;
import com.poscoict.glueframework.context.GlueContext;

public class #CLASS# extends GlueActivity<GlueContext> {
    // Do not define the Class Member Variable!!
    public String runActivity(GlueContext ctx) {
        this.logger.info("test");

        return GlueBizControlConstants.SUCCESS;
    }
}

]]>
    </activity-template>
</common>
</glue-eclipse-config>

```

18.2 applicationContext.xml

Glue Framework 은 spring 기반의 개발프레임워크이며, application 개발시 applicationContext.xml 을 항상 필요로 한다. 그리고 applicationContext.xml 에는 serviceManager bean 을 포함한다.

다음은 serviceManager 를 갖고 있는 applicationContext.xml 의 내용이다. xsd 기반의 applicationContext.xml 을 사용하도록 한다.

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id="serviceManager"
        class="com.poscoict.glueframework.biz.control.GlueServiceManagerImpl"
        lazy-init="true">
        <property name="cacheManager" ref="cacheManager" />
    </bean>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

    <property name="serviceLoader" ref="serviceLoader" />
</bean>

<bean id="serviceLoader"
      class="com.poscoict.glueframework.biz.control.GlueServiceLoader"/>

<bean id="cacheManager"
      class="com.poscoict.glueframework.cache.ehcache.GlueEhCacheManager"/>
<!-- EhCache : com.poscoict.glueframework.cache.ehcache.GlueEhCacheManager -->
<!-- JCS      : com.poscoict.glueframework.cache.jcs.GlueJCSCacheManager -->
</beans>

```

다음은 dtd 를 사용한 applicationContext.xml 의 내용이다. xsd 기반과 동일하나, 확장성을 위해 권장하지 않는다.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans>

    <bean id="serviceManager"
          class="com.poscoict.glueframework.biz.control.GlueServiceManagerImpl"
          lazy-init="true">
        <property name="cacheManager" ref="cacheManager" />
        <property name="serviceLoader" ref="serviceLoader" />
    </bean>
    . . . 중략 . . .
</beans>

```

18.3 web.xml

web application 에서 web.xml 이라 불리는 deployment descriptor 는 WEB-INF 디렉토리에 위치해야 한다.

다음은 Eclipse IDE(Eclipse Java EE IDE for Web Developers, Kepler 버전)를 통해 servlet spec 별로 생성되는 default web.xml 이다.

Servlet Spec 3.0 일 경우

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns="http://java.sun.com/xml/ns/javaee"
          xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                              http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
          id="WebApp_ID"
          version="3.0">
    <display-name>sample</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
    </welcome-file-list>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>

```

Servlet Spec 2.5 일 경우

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  id="WebApp_ID"
  version="2.5">
  <display-name>test</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>

```

Servlet Spec 2.4 일 경우

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID"
  version="2.4"
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>test</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>

```

Servlet Spec 2.3 일 경우

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app id="WebApp_ID">
  <display-name>test</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
</web-app>

```

Glue Framework에서는 SpringMVC 와 Struts 를 선택해서 사용할 수 있으며, web.xml 에 DispatcherServlet 또는 ActionServlet 을 추가해야 한다.

18.3.1 Spring MVC

Spring MVC 를 사용하는 경우의 web.xml 은 다음과 같은 형태를 활용 한다.

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">

    <description>sample</description>

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/dispatcher-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>*.mvc</url-pattern>
    </servlet-mapping>

</web-app>

```

18.3.2 Struts 1

Struts 1 을 사용하는 경우의 web.xml 은 다음과 같은 형태를 활용 한다.

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">

```

	문 서 명	Glue Framework 개발자 가이드
	제품버전	V 4.0

```

<description>sample</description>

<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>

</web-app>

```

posco ict